12 Introduction to software processes

- Study of software processes is the core of software engineering
  - successful projects of the past
    - the processes that worked well
    - a prescription for the future projects
  - unsuccessful projects
    - find the problems that led to the failure
- Good process $\rightarrow$ Good product

Variability of processes

- Team
  - organization, collaboration, skills
- System
  - technology, domain, size, and expected quality

Granularity

- Coarse granularity processes
  - deal with long periods of time
  - software life span models are example of a process of very coarse granularity
  - stages that are also processes
- Word “process” usually used for processes that fit within a single stage or few neighboring stages

Granularities of processes

Granularity	Example
lifecycle	staged, waterfall
stage	evolution, servicing
process	SIP, AIP, DIP
task	software change, acceptance testing
subtask, phase	concept location, actualization
step, action	inspection of a class

Forms of process

- Process model
  - prescription what the tasks should be and how should they fit together
  - a blueprint how to do things
- Enactment
  - the actual process in the project
  - inevitable deviations and exceptions from the process model

Forms of process (2)

- Performance
  - set of measures that an observer of an enacted process collects
    - time, cost, quality, . . .
- Plan
  - expected future performance
  - decisions that the project stakeholders take
    - alternatives how to enact the process model

Solo iterative process (SIP)

- Single programmer repeats software changes
- Functionality is added one step at a time
- Repeated changes are the basis of
  - software evolution
  - software servicing, reengineering
- SIP demonstrates characteristics shared by all iterative processes

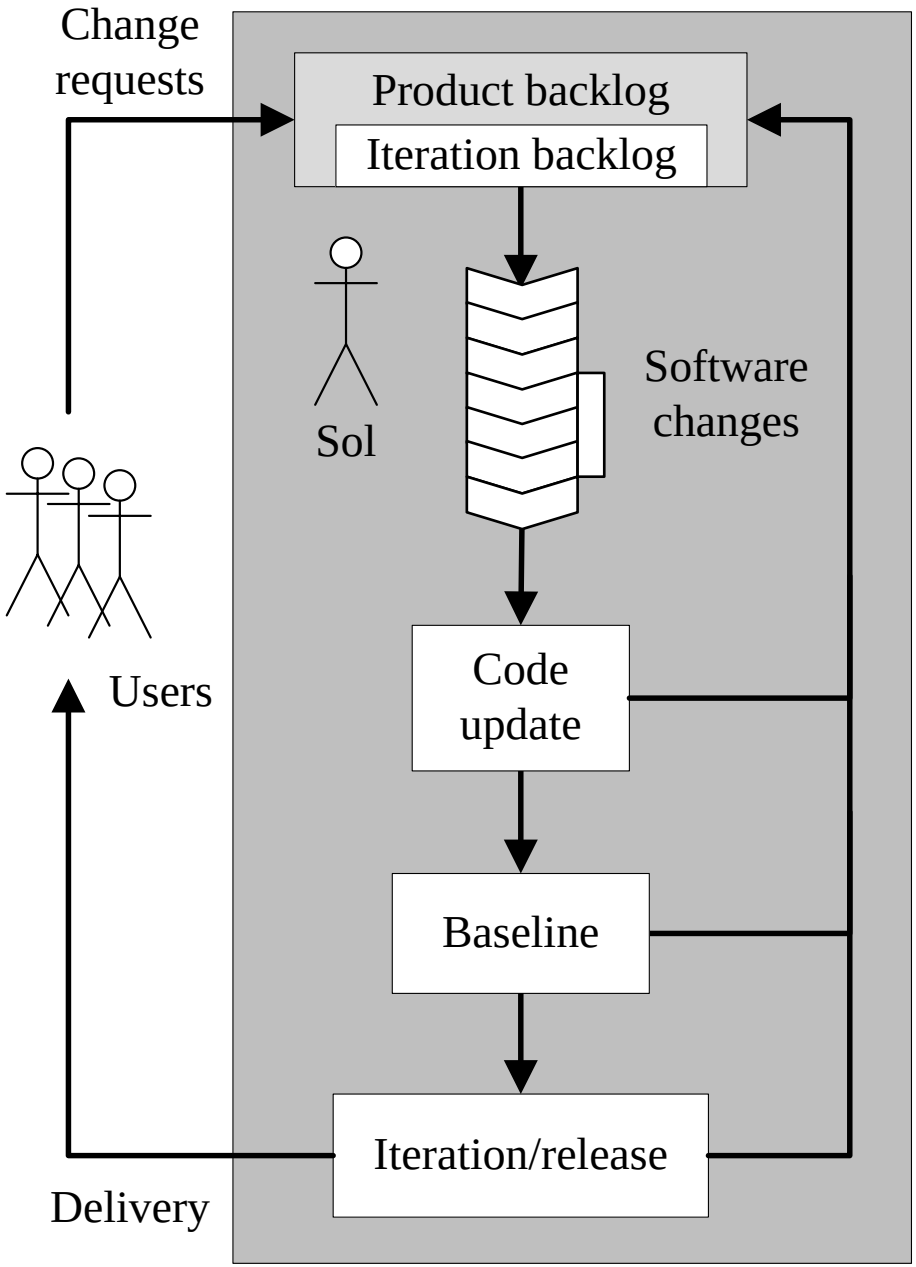
Why SIP at all?

- Why should solo programmer follow a predefined process?
  - rather than react flexibly to challenges
- Answer: even the solo programmers have to meet their obligations
  - fulfill their promises, pay their bills
  - plan the future
  - manage their own resources
- SIP is the process that allows that

Workproducts

- Product backlog
  - change requests
  - represents the vision for the future of the software.
  - bugs in the functioning of software
  - new demands and ideas
- Software code
- Software documentation, etc.

SIP model


Change requests
Product backlog
Iteration backlog
Sol
Software changes
Users
Code update
Baseline
Iteration/release
Delivery

Enactment of SIP

Task		Comments
1	Pri	New change request arrives
2	Ini	Add Cashier Session
3	CL	CashierRecord
4	IA	Estimated set has 4 classes
5	Ref	Extract class Session
6	Ex	Install new version of Bugzilla
7	Act	Replace class Session
8	Base	2 regression faults added to backlog
		. . .

Measuring SIP

- Data indicate how the process is working
- Data serve as a foundation for future planning

Time log

Start	End	Interruption		Process enactment	
		#	time	Step	Comments
8:23	8:31			Pri	
8:32	8:39	1	2	Ini	Add Cashier Session
8:42	8:52			CL	CashierRecord
8:52	9:23	2	12	IA	4 classes
9:27	10:46	3	25	Ref	extract Session
10:50	11:42	2	6	Ex	Downloading and installing new version of Bugzilla
1:23	2:17			Act	class Session
2:22	3:12	3	12	Base	2 regression faults added to the backlog

Time

- Total time
- Clean time
  - American football game
  - 15 minutes is the clean time of each quarter
  - total time includes all interruptions
- Clean time = end – start – time of interruptions

Log = raw data

- Can become large
- From the log the following can be extracted
  - weekly summary of the clean time
  - average time for concept location
  - concept location is becoming faster or slower
  - recurring exception to the SIP
  - . . .

Program size

- Number of lines of source code
  - LOC, KLOC, MLOC
- This measure is very inaccurate
  - different programming languages=different size
  - different programming styles=different size

LOC

- LOC is the most commonly used measure of program size
- Only the one or two most significant digits are meaningful
  - 900 LOC, 23 KLOC, 3.2 MLOC

Other measures

- Function points
  - correlate with LOC, harder to compute
- Measures even less accurate than LOC
  - number of methods in the source code
  - number of classes
  - number of files

Code defects

- Incorrect computations, premature termination
- Defect density
  - good quality software: ~2.0 defects per KLOC
  - poor quality software: Higher density
  - avionics software
    - defect density estimated 0.1 defects per KLOC
    - cutting edge of what can be achieved
    - NASA Space Shuttle

Defect log

	Defect Found			Location	Description	Origin		Fixed
	Date	Time	Task			Date	Task	
1	11/4	9:00	CL	Cashier.get()	for I = 0, loop does not terminate	3/12	Act	12/8
2	11/4	2:32	Base	--	The pop-up window for 3rd cashier does not appear	?		11/25
3	11/4	3:02	Base	Price.get()	Price = 0 raises exceptions	4/21	Ref	
4				. . .				

Planning

- Planning is a prediction of the future
  - there are uncertainties and risks involved
- Data about the past are good predictors about the future
  - recording the past and planning the future are closely related
  - future cannot be predicted with any level of certainty without knowing the past.

Repetitions

- Easiest things to predict are the repetitions
- Eliminating the risk and uncertainty is one of the main topics of the planning.
  - emphasize the repetitive nature of the process
  - “repetition is the mother of skill”
- Unique and unprecedented tasks are hard to plan

Planning software changes

- Analogy
  - estimate the time needed for the phases
    - find similar phases in the past, use their numbers as the basis
- Decomposition
  - decompose the change into phases
  - get the sum for all phases
  - errors may compensate each other

Tasking

- Changes should be made more alike
  - they will be more predictable
- Narrow range of size
  - “epics”
  - divide large changes into smaller ones

Tasking example

- “Customers can download and then use sales coupons online”
- Subtasks
  - build a web site for the store and coupons
  - support the store manager to create and remove coupons
  - customer payment involves cashing coupons
  - database that stores how many coupons were used

Baselines

- Schedule them at regular intervals
  - every day at the end of the shift
  - every other day after a change is finished
- Postponing is not recommended

Release plan

- Involve business considerations
- Release with a certain new functionality on a certain date
  - planning makes sure that this promise is realistic.

Release backlog table (original)

Plan after x hours of work	0
1: initial	10
2: inventory	30
3: multiple prices	30
4: promo prices	30
5: cashier login	20
6: multiple cashiers	30
7: cashier sessions	35
8: detailed sale	30
9: multiple line items	35
10: payment	20
11: credit payment	40
12: check payment	20
remaining effort	330
total effort needed to reach the goal	330

Release backlog table after 100 hours of work

Plan after x hours of work	0	100
1: initial	10	10
2: inventory	30	50
3: multiple prices	30	40
4: promo prices	30	30
5: cashier login	20	20
6: multiple cashiers	30	30
7: cashier sessions	35	80
8: detailed sale	30	30
9: multiple line items	35	70
10: payment	20	20
11: credit payment	40	40
12: check payment	20	20
remaining effort	330	340
total effort needed to reach the goal	330	440

Release backlog table (late release)

Plan after x hours of work	0	100	285	405	475
1: initial	10	10	10	10	10
2: inventory	30	50	50	50	50
3: multiple prices	30	40	40	40	40
4: promo prices	30	30	30	30	30
5: cashier login	20	20	20	20	20
6: multiple cashiers	30	30	55	55	55
7: cashier sessions	35	80	80	80	80
8: detailed sale	30	30	30	30	30
9: multiple line items	35	70	70	70	70
10: payment	20	20	20	20	20
11: credit payment	40	40	40	50	50
12: check payment	20	20	20	20	20
remaining effort	330	340	180	70	0
total effort needed to reach the goal	330	440	465	475	475

Release backlog table (incomplete release)

Plan after x hours	0	100	285	405
1: initial	10	10	10	10
2: inventory	30	50	50	50
3: multiple prices	30	40	40	40
4: promo prices	30	30	30	30
5: cashier login	20	20	20	20
6: multiple cashiers	30	30	55	55
7: cashier sessions	35	80	80	80
8: detailed sale	30	30	30	30
9: multiple line items	35	70	70	70
10: payment	20	20	20	20
11: credit payment	40	40	40	50
12: check payment	20	20	20	20
remaining effort	330	340	180	70
total effort needed to reach the goal	330	440	465	475