Evolutionary Development Methods (Evo)

Simon Porro
SPI Partners BV
+31- 40 - 248 98 22
porro@spipartners.nl
www.spipartners.nl

Niels Malotaux
N R Malotaux - Consultancy
+31- 30 - 228 88 68
niels@malotaux.nl
www.malotaux.nl/nrm/English

Agenda

- **Part One - EVO Basics (40 min)**
  - **Evo principles**
  - **Evo compared to XP**
  - **Evo and CMM(I)**
- **Part Two - Managing Projects with EVO (40 min)**
  - **Task & Delivery Cycles**
  - **How to turn a project into an Evo Project**
  - **Results**

Simon Porro

- Computing Science 1981 - 1987
- Software Development, project Leader, Group Leader, Quality Consultant
- Since 1995 SPI Consultant, CMM, CMMI, ISO 9000-3, EFQM, PQA, BEST
- Current activities: training & coaching
  - Evolutionary Project organisation (Evo)
  - Requirements & Strategic Objectives Specification
  - Project Rescue
  - Reviews and Inspections
  - CMM, CMMI Training, Assessments & Consulting

Development Goals

- **The right product**
- **The right quality**
- **Within the time and budget agreed**
- **Pleasant for everyone involved**

Quality On Time

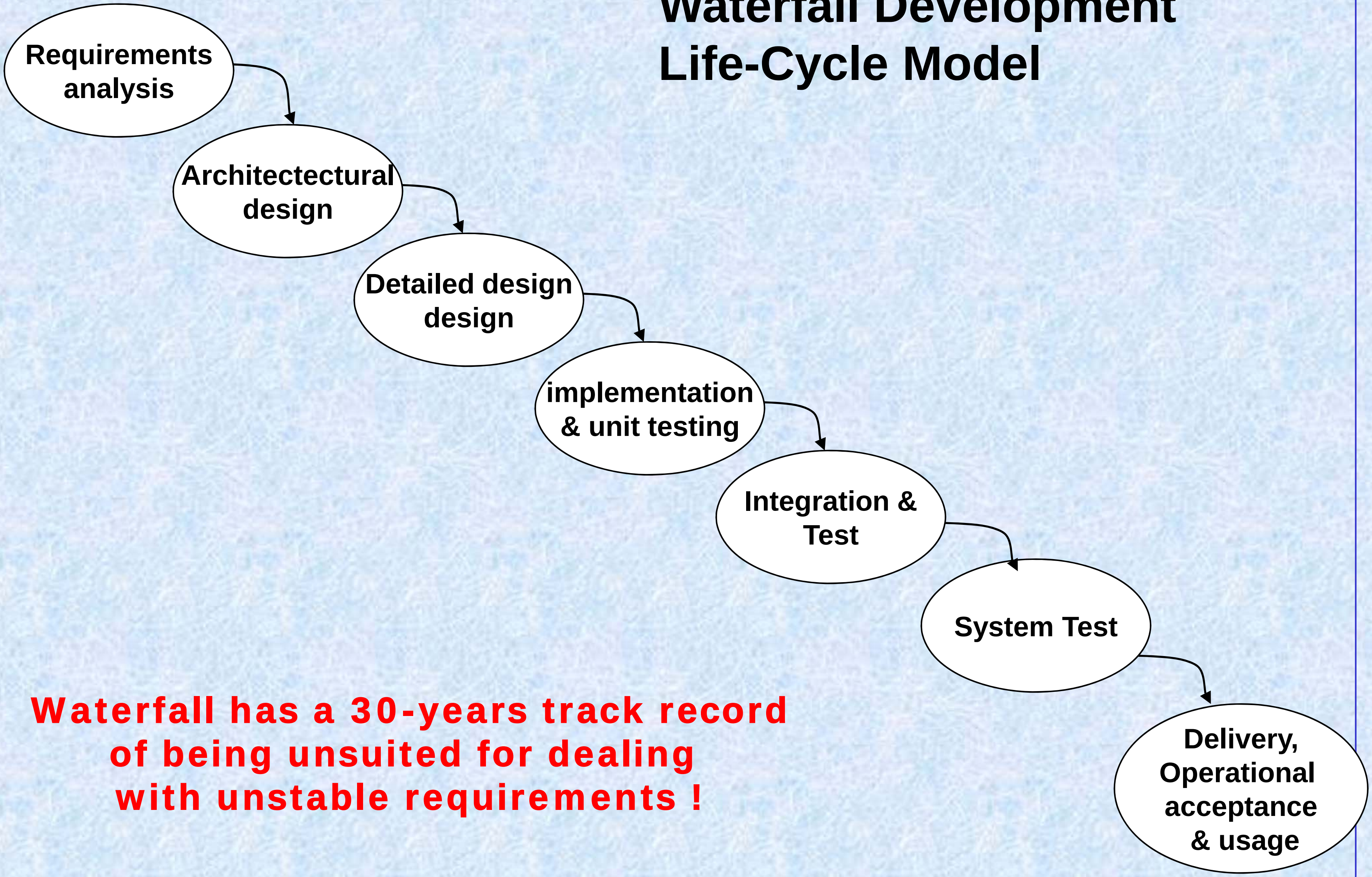
The Requirements Paradox

- Requirements must be stable
- Requirements always change

→ Use a process that can cope with the requirements paradox

You cannot foresee every change,
but you can foresee change itself

Waterfall Development Life-Cycle Model

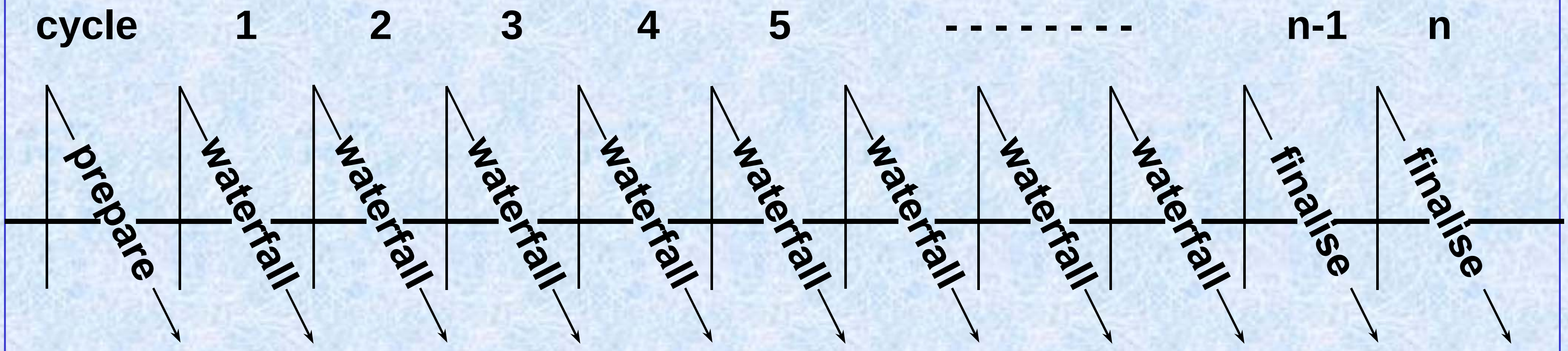


Waterfall has a 30-years track record of being unsuited for dealing with unstable requirements !

The 2nd Requirements Paradox

- **We don’t want requirements to change**
- **Because requirements change is a *known risk:***
  We must *provoke* requirements change
  as early as possible

Evo is many waterfalls/V-models



Requirements Analysis	Design Engineering	Construction/Acquisition	Test (System, Acceptance)

Waterfall development model (Big Bang delivery)

Complete Detailed Frozen	Complete Detailed Frozen	Build/test	Build/test	Build/test	Build/test	Build/test	Deliver
Requirements Analysis & specification	**Design Specification**	**Step 1** →	**Step 2** →	**Step 3** →	**Step 4** →	**Step n** →	**Contract Acceptance Test**

Incremental development model (technical selection of increments)

Best guess Updated stepwise	Best Guess Updated stepwise	Reqs Design Build Test Deliver	Feedback/ Reqs Design Build Test Deliver	Feedback/ Reqs Design Build Test Deliver	Feedback/ Reqs Design Build Test Deliver	Feedback/ Reqs Design Build Test Deliver	
Requirements Analysis & specification (needs)	**Design specs** **(ideas)**	**Step 1** →	**Step 2** →	**Step 3** →	**Step 4** →	**Step '50'** →	**Contract Acceptance Test**

Evolutionary development model (stakeholder value selection of iterations)

Ref. Tom Gilb: Evo

EVO Principles

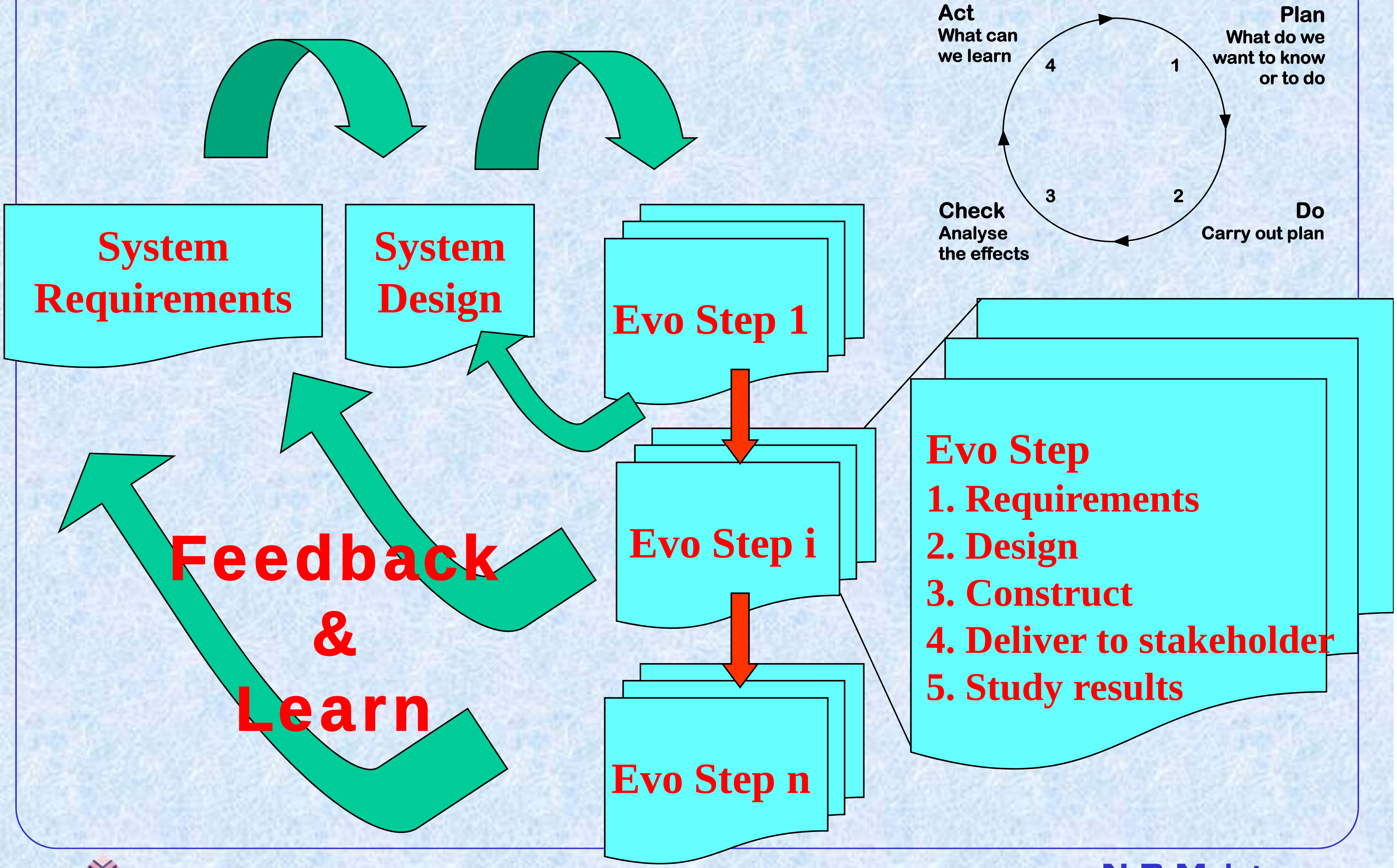
1. **Very frequent, early value delivery** to stakeholders
   - weekly cycles, 2% of project budget
2. **Rapid feedback** from stakeholders on delivered values
3. **Most juicy/risky/critical stakeholder values** are delivered first
4. **Multi-disciplinary development teams**

5. **Quantification** of all critical stakeholder values using Planguage:
   - Requirements defined on a Scale of Measure
   - Target stakeholder value levels: Must, Plan, Wish
6. **Dynamic Prioritization**

   The exact content of next week's EVO delivery cycle is based on:
   - The current planning
   - This week's cycle results
   - Changed requirements and priorities
   - Feedback from stakeholders

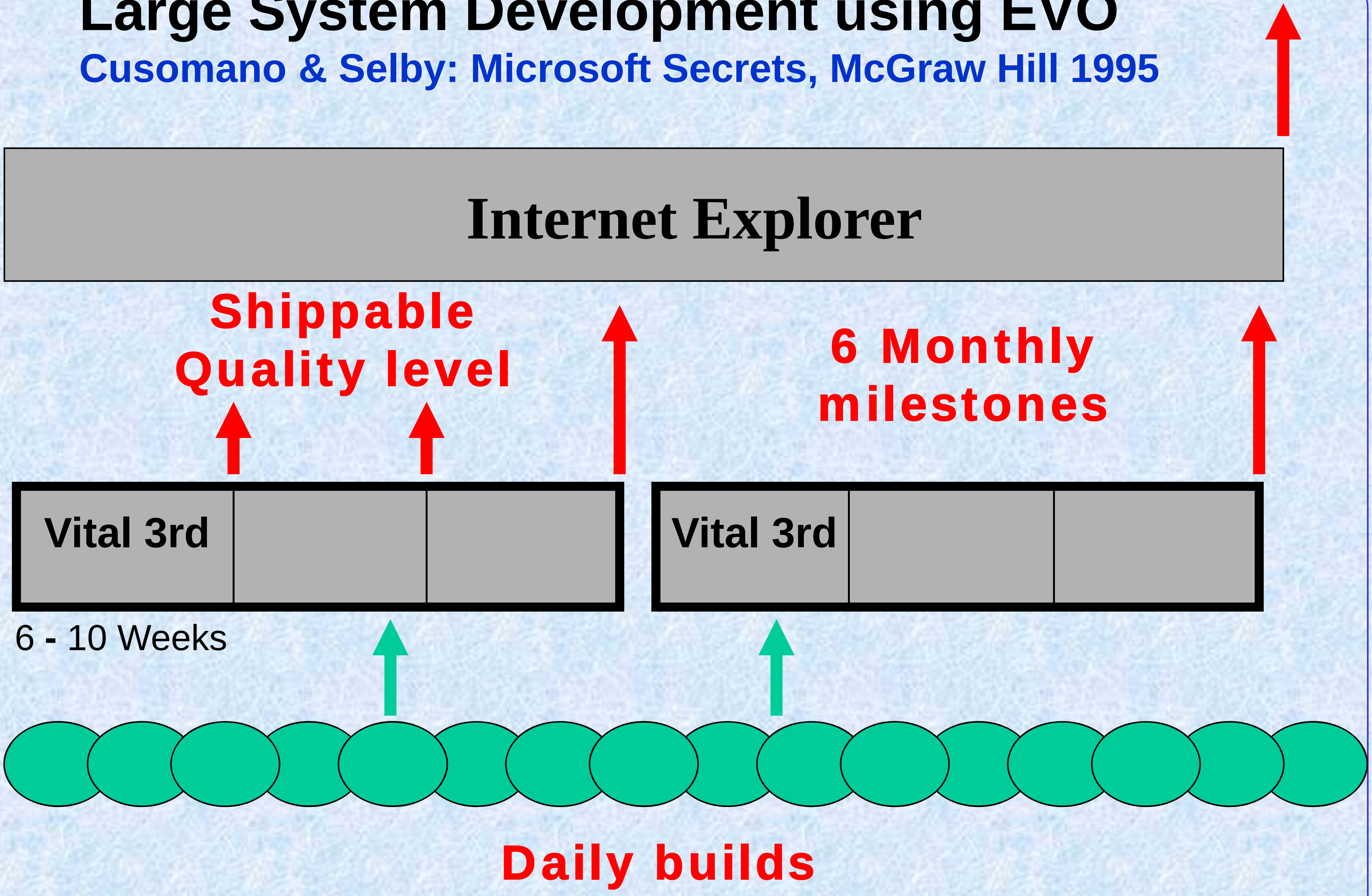
In chess, your next move is based on the board situation and your opponent's last move

Evo ‘Learning’ through Feedback



Large System Development using EVO

Cusomano & Selby: Microsoft Secrets, McGraw Hill 1995

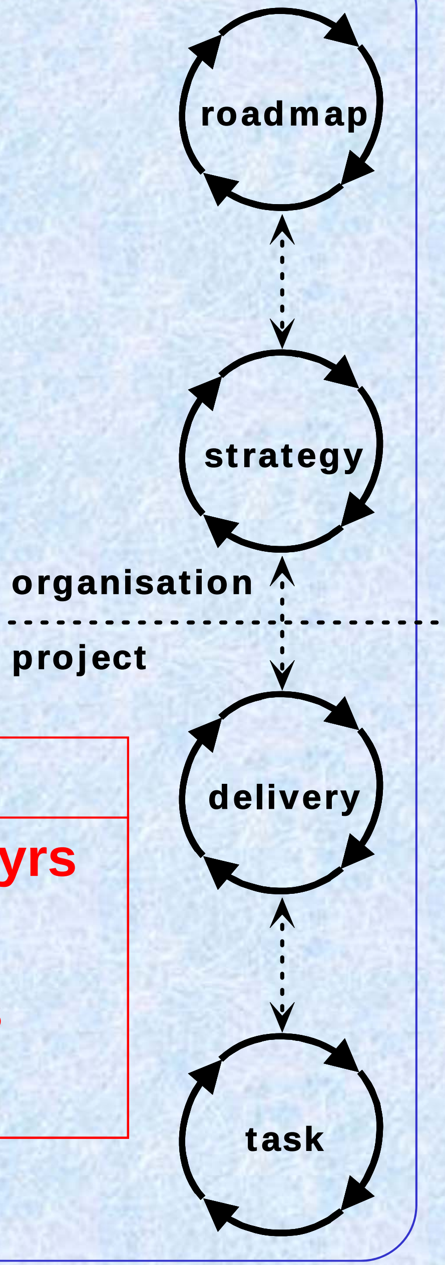


EVO Management:

Which roles are involved in the EVO Team?

	PL	RE/ Arch	Dev Team	Lib	Test Eng.	CS Eng.	Stakeh. PM, Beta Site
One EVO Delivery Cycle includes:							
- Weekly Evaluation	X	X	X	X	X	X	X
- (MS-1) Step Planning	X	X	X	X	X	X	
- Requirements	X	X				X	X
- Design		X	X				
- Test Design		X			X		
- Check-out			X	X			
- Coding			X				
- Unit-test			X				
- Check-in			X	X			
- Integration with existing system				X			
- Integration & regression test (MS-7)	X	X	X	X	X	X	
- Possibly:							
- System Test (MS-8)	X		X		X	X	
- (Restr.) Delivery to Stakeholder	X			X	X	X	X

Cycle-types in Evo



	Frequency	Horizon
Roadmapping Cycle	3 - 6 mo	6 mo - 2 yrs
Strategic Objectives Cycle	1 mo	3 - 6 mo
Value Delivery Cycle	1 - 2 wks	1 - 8 wks
Task Cycle	≤ 1 wk	

Functional and Quality Requirements

- 90% of all requirements are functional requirements (features)
- Most functional requirements are really designs
- Most functional requirements have undocumented underlying requirements. Just ask: “why do you want this feature?”
- The underlying requirements (strategic objectives) are often qualitative by nature
- All Qualitative Requirements can always be specified on a Scale of Measure
- Quantifying the Strategic Objectives of a project brings very strong focus on results

Example: Strategic Objectives.OSW.[Product]

- **Synchronization** (of [XXX] Software with Assembleon products)
- **Machine-Line Utilization Effectiveness** (% maximum)
- **Functional Accuracy**
- **Performance** (execution speed)
- **Usability**
  - **Learnability**
- **Serviceability** (how fast we can ‘service’)
- **Availability** (uptime / failure rate)
  - **Reliability**
  - **Maintainability (how fast we ‘repair’ faults)**
- **Security**
- **Quality of Product Information** (to Stakeholders)
- **Accessibility**
- **Adaptability**

Planguage Example: Quantifying Goals:
Product Synchronization

- **Ambition:** [Product] is never late for delivering needed and promised software to support Assembleon products releases
- **Stakeholder:** {Assembleon Sales, Assembleon CEO, other Product Teams, Customers, Prospects}
- **Scale:** Days Late compared to published or agreed delivery date
  - Days Late: Defined As: Calendar Days between agreed/promised delivery dates and the first whole day when Correctly Installed and Really Available for Customer Use, including all Necessary training, support and documentation

====Benchmarks ============= the Past

- **Past** [Emerald FNC, 2000, Optimiser] 5 months late ← FvL

===== Targets ============= the Future

- **Must** [GEM, During 2001] 1 month late ← Product Manager
- **Plan** [All Products, 2001] 15 days
- **Wish** [All OSW Products, Q4 2001] 0 days or better ← ALL OF US

Example: Quantified Priority Setting

‘Impact Estimation’

Selection Values (below)	Alternatives →	Strategy 1 / Design 1	Strategy 2 / Design 2	
Synchronization		3	9	0 = no value
Reliability		8	2	9 = top value
Machine Utilization		8	0	
Timing Accuracy		9	0	
Usability		2	9	
-------	COSTS	-------	-------	
Engineer Hours		300	40	
Value/Cost ratio		.10	.50	

Impact Table for Cycle Planning & Evaluation

	Step #1 Plan A: {Design-X, Function-Y}	Step #1 Actual	Difference. - is bad + is good	Total Step 1	Step #2 Plan B: {Design Z, Design F}	Step #2 Actual	Step #2 Difference	Total Step 1+2	Step #3 Next step plan
Reliability 99%-99.9%	50% ±50%	40%	-10%	40%	30% ±20%	20%	-10%	60%	0%
Performance 11sec.-1 sec.	80% ±40%	40%	-40	40	30% ±50%	30%	0	70%	30%
Usability 30 min. -30 sec.	10% ±20%	12%	+2%	12%	20% ±15%	5%	-15%	17%	83%
Capital Cost 1 mill.	20% ±1%	10%	+10%	10%	5% ±2%	10%	-5%	20%	5%
Engineering Hours 10,000	2% ±1%	4%	-2%	4%	10% ±2.5%	3%	+7%	7%	5%
Calendar Time	1 week	2 weeks	-1week	2 weeks	1 week	0.5 weeks	+0.5 wk	2.5 weeks	1 week

Managerial Consequences of EVO Implementation

- More frequent communication with the stakeholders
- More integration effort (more CM)
- Project needs Requirements Engineer & Architect during the entire project
- More intensive priority setting and scheduling for the project leader (which he should have done in the first place)

EVO can very well be combined with existing PCP processes.

Don’ t use EVO as excuse for abandoning other useful project management and PCP practices!

How does EVO affect CMM(I) compliance? → Level 2

- **RM:** EVO strongly supports RM.
- **PP:** Keep existing overall estimating techniques for size, complexity, effort and CCR. Schedule according to dynamic EVO priorities.
- **PTO:** EVO = continuous tracking & correction of plans.
  Do not abandon existing management reporting procedures
- **SM:** Applying EVO-principles to the subcontractor reduces risk
- **SQA:** Very frequent review & testing (QC), Independent QA must be covered separately
- **SCM:** Just apply all existing CM procedures (more integration cycles).
- **M&A:** Well implemented EVO provides weekly product completion & quality measures. Process Performance Measurement must be added.

How does EVO affect CMM(I) compliance? → Levels 3, 4

- **IC:** **EVO provides active synchronisation with other groups and disciplines: some support for IC.**
- **SQM:** **Quality attributes are numerically specified. Their scales of measure form a good entry for applying statistical process control.**

Overlaps between Evo and XP (BLUE)

Planning

- User stories are written
- Release planning creates the schedule
- **Make frequent small releases**
- **The Project Velocity is measured**
- **The project is divided into iterations**
- **Iteration planning starts each iteration**
- Move people around
- A stand-up meeting starts each day
- **Fix XP when it breaks**

Designing

- **Simplicity**
- Choose a system metaphor
- Use CRC cards for design sessions
- **Create spike solutions to reduce risk**
- **No functionality is added early**
- **Refactor whenever and wherever possible**

Coding

- The customer is always available.
- **Code must be written to agreed standards.**
- Code the unit test first.
- All production code is pair programmed.
- Only one pair integrates code at a time.
- **Integrate often.**
- Use collective code ownership.
- Leave optimization till last.
- **No overtime.**

Testing

- All code must have unit tests.
- All code must pass all unit tests before it
- can be released.
- When a bug is found tests are created.
- **Acceptance tests are run often and the score is published.**

Differences between Evo and XP

EVO

- Suited for large & small Systems & Software Development
- Results Centric
- Stakeholder focus
- Works with anybody
- Numeric
  - specifiaction of (strategic) objectives
  - prioritization (impact tables)
  - progress tracking

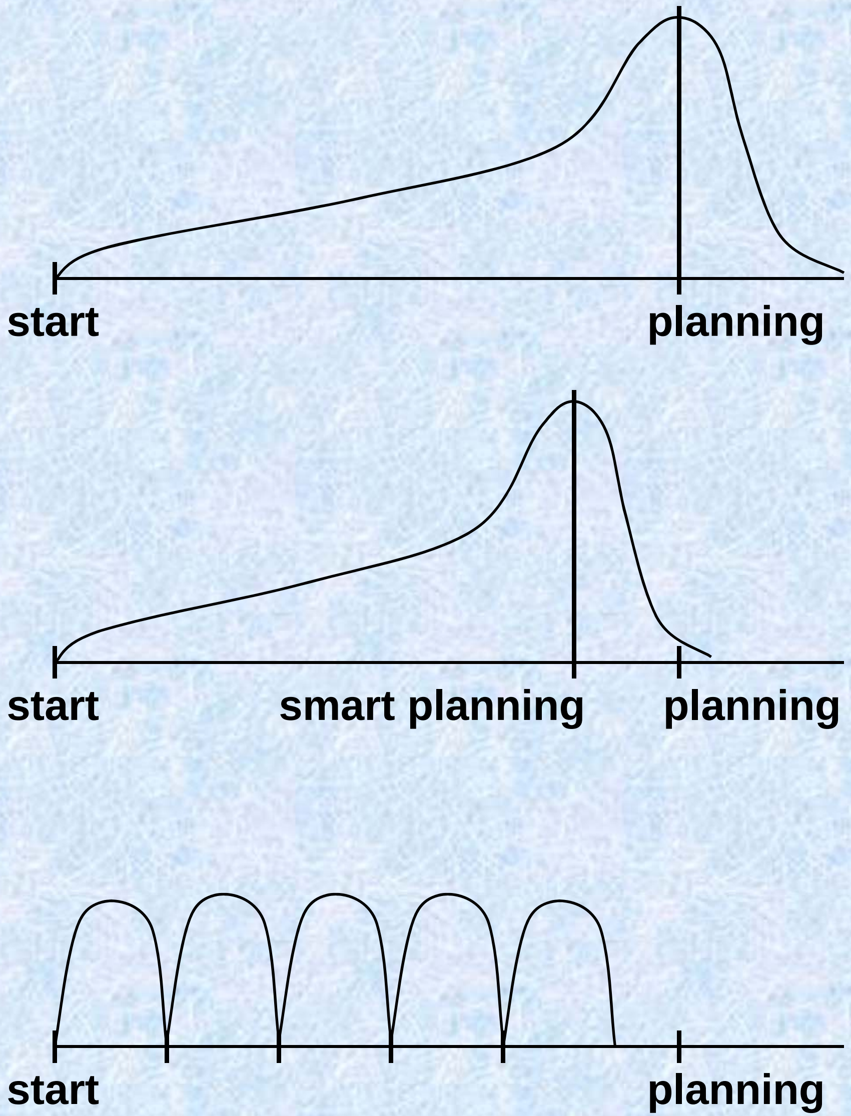
XP

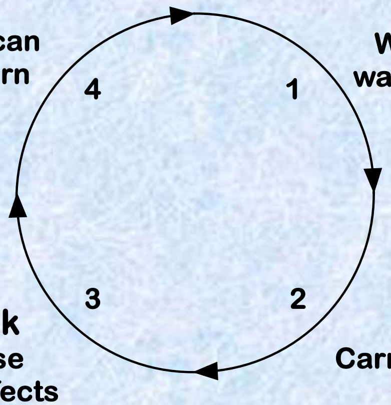
- Suited for small Software Development only
- Code Centric
- Developers focus above Process focus
- Need seasoned programmers
- NO numeric specification of objectives, prioritization nor tracking

Niels Malotaux

- Electronics 1974
- Development of computers, embedded systems and software
- Since 1998 “Quality On Time” consultant
  - Optimising outsourcing
  - Optimising way of working R&D organisation
  - Optimising way of working software organisation
- Current activities: training & coaching
  - Evolutionary Project organisation (Evo)
  - Requirements engineering
  - Reviews and Inspections
  - Project Rescue

Development cycles





Act
What can
we learn

Plan
What do we
want to know
or to do

Check
Analyse
the effects

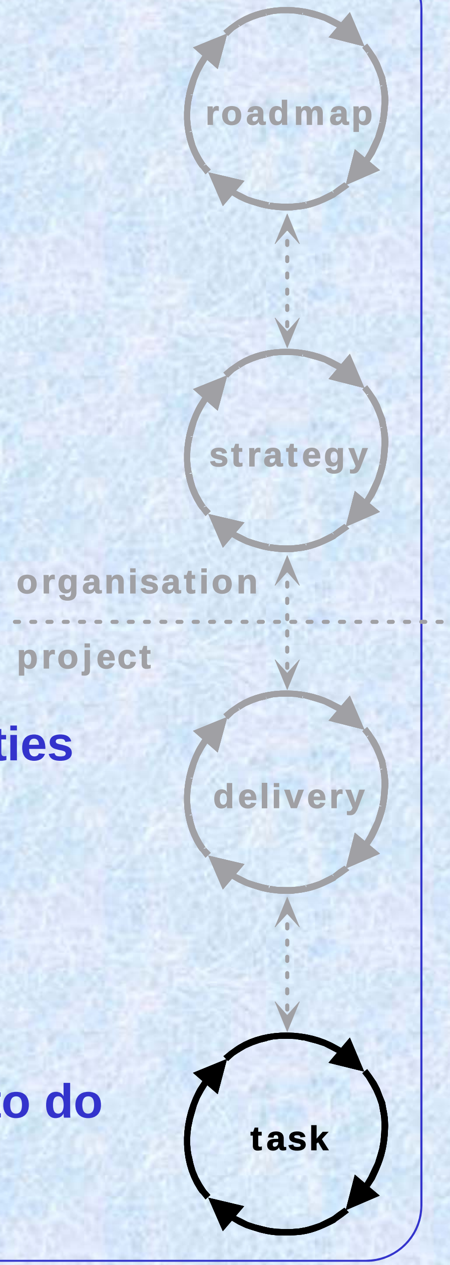
Do
Carry out plan

Discipline

- Control of wrong inclinations
- Discipline is very difficult
- We must help each other

Romans 7:19

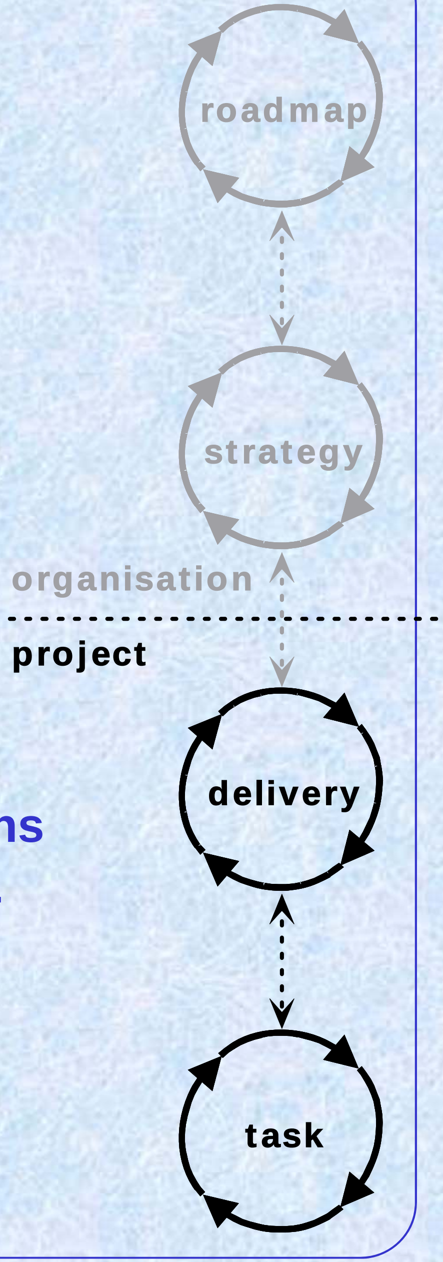
Cycles in Evo



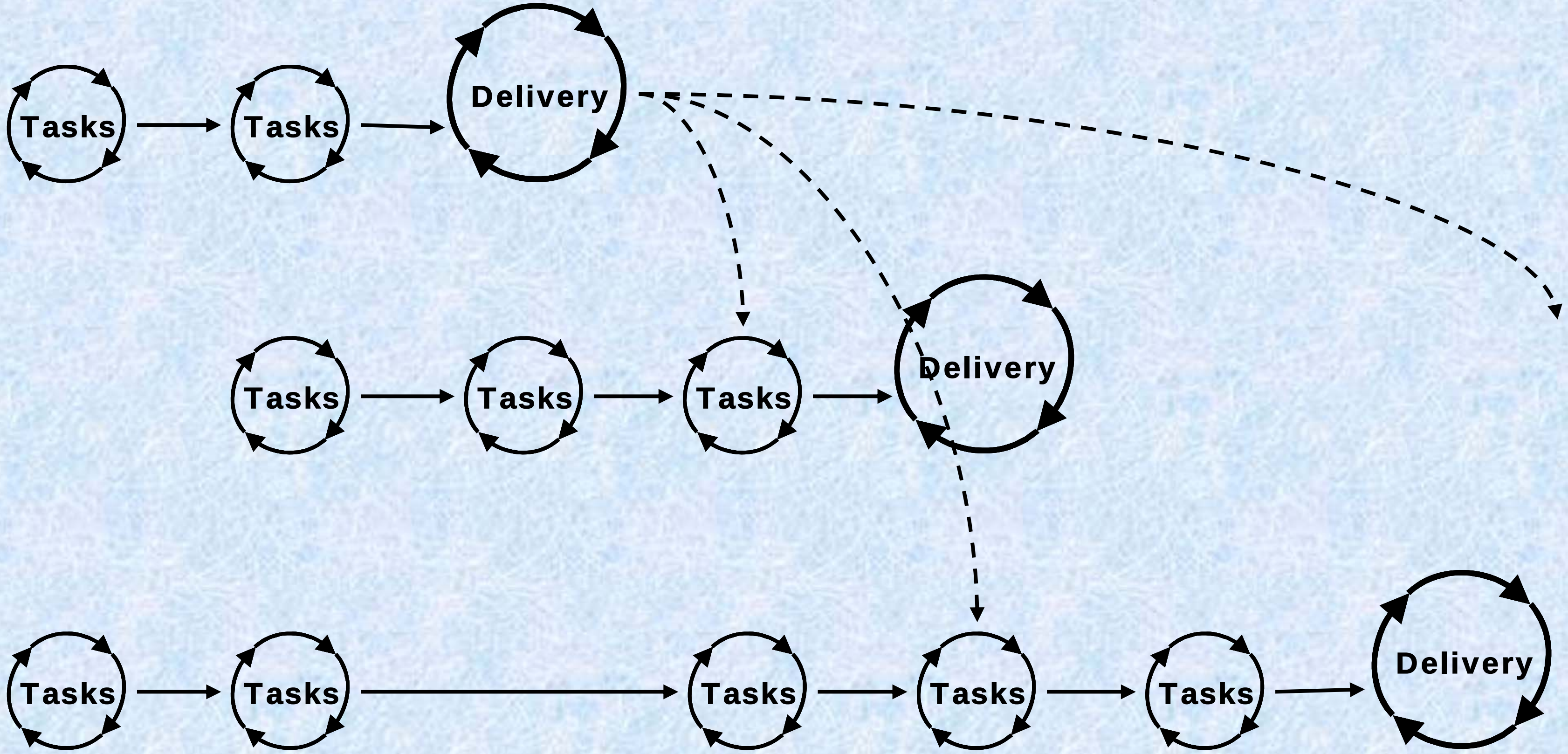
- **Weekly Task Cycle**
  - **Are we *doing* the *right things*, in the *right order,* to the right *level of detail***
  - **Optimising estimation, planning and tracking abilities to better predict the future**
  - **Select highest priority tasks, never do any lower priority tasks, never do undefined tasks**
  - **There are only about 26 real effort hours in a week**
  - **In the remaining time: do whatever else you have to do**
  - **Tasks are always done, 100% done**

Cycles in Evo

- **Weekly Task Cycle**
- **Value Delivery Cycle**
  - **Are we *delivering* the *right things*, in the *right order*, to the right *level of detail***
  - **Optimising requirements and checking assumptions**
  - **Delivering the juiciest, most important stakeholder values that can be made in the least time**
  - **1 to 2 weekly cycles**



Tasks feed deliveries



Task Cycle - Delivery Cycle

Doing **Delivering**

the *right things*, in the *right order* to the *right level of detail*

Optimising

Estimation, planning, tracking	**Requirements, assumptions**

Selecting

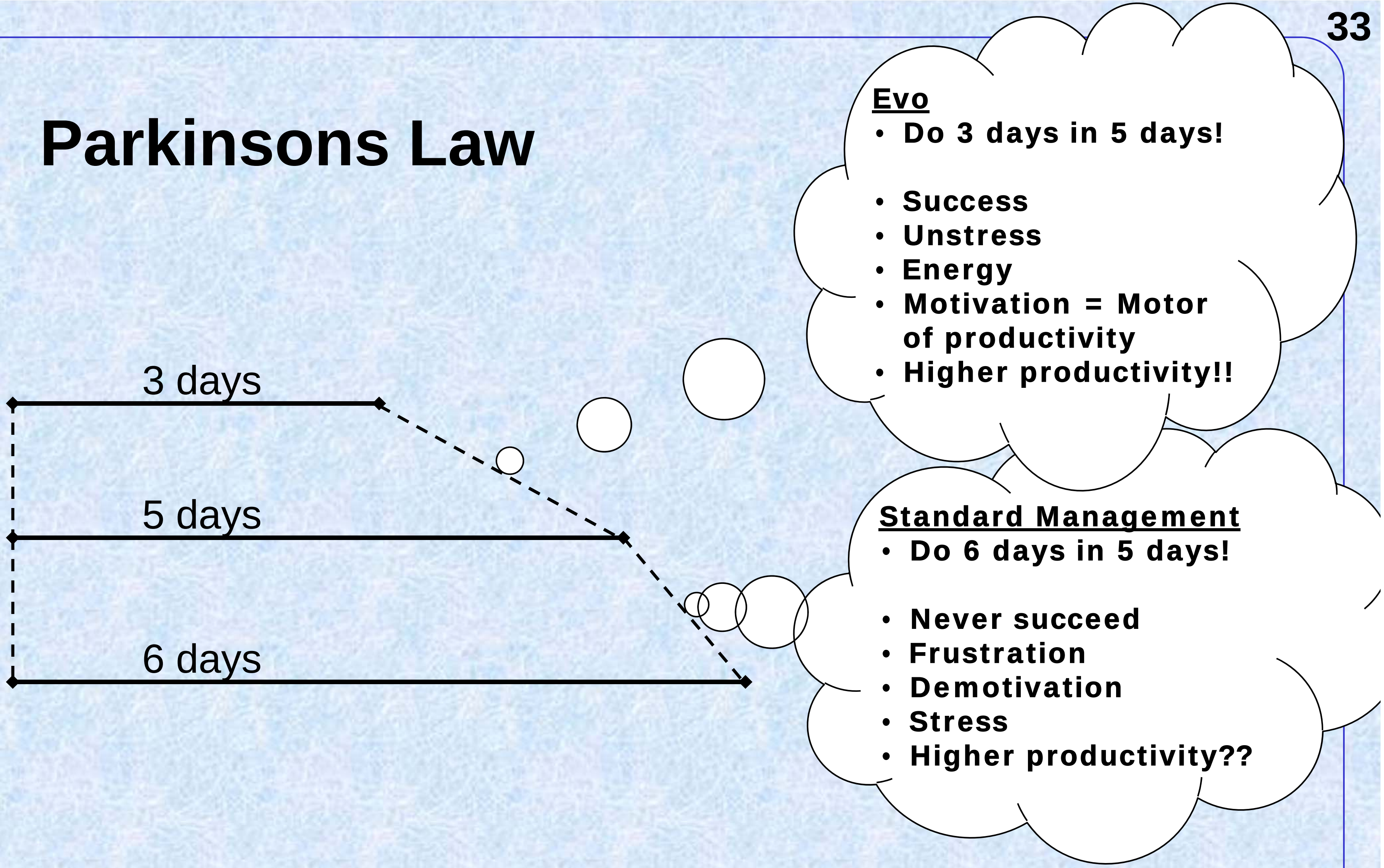
Highest priority tasks	**Juiciest, most important values**
≤ 1 week	**1 to 2 task cycles**

Always done, 100% done

How to start with tasks

- **Take the requirements, architecture and design**
- **Make a list of things to do**
- **Split in tasks of 26 hours max (use *effort* estimation)**
- **Put on List of Candidate tasks**
- **Prioritise the tasks on the Candidate List**
- **Select ~26 hrs of tasks from top of the list**
- **Agree and commit to work packages (100% done!!!)**
- **Use TaskSheets to avoid extra work (what, how, how check, how done)**
- **Do the work**
- **Learn**

Parkinsons Law



Evo

- **Do 3 days in 5 days!**

- **Success**
- **Unstress**
- **Energy**
- **Motivation = Motor of productivity**
- **Higher productivity!!**

Standard Management

- **Do 6 days in 5 days!**

- **Never succeed**
- **Frustration**
- **Demotivation**
- **Stress**
- **Higher productivity??**

“Work expands to fill the time available”

Evo Day: Goal

Turning a project into an Evo project

At the end of the day:

- Everyone knows what to do and why in the next cycle
- 100% commitment given
- We know that we are going to work on highest priority issues

Evo Day: Morning

- Presentation of Evo Methods
  - Like this story
- Presentation of product
  - How well do we know the goals of the project?

Evo Day: Afternoon

- Decomposing work into subtasks (of max 26 hours effort)
  - Estimate effort in hours
  - Estimate priority
  - Who could best do this
- Listing tasks in order of priority
  - How to define priority order
- Top of the list (highest priority issues):
  - Estimate is not yet done
  - Who should do what
  - Take your tasks from the list for coming cycle (week)
  - Commit to finish these tasks completely

Task selection criteria

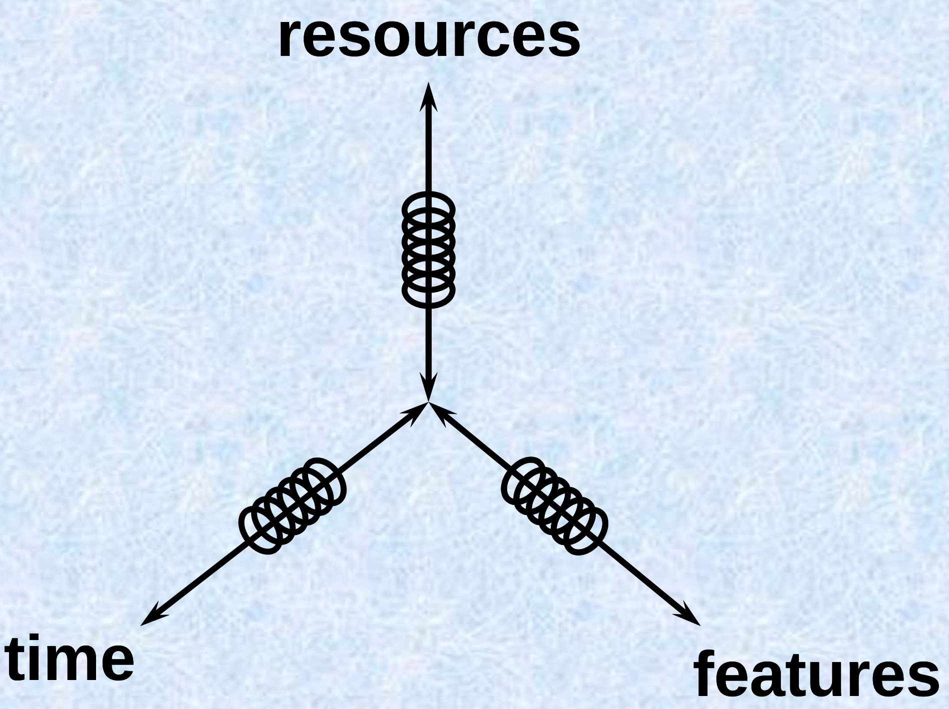
- **Most important requirements first**
- **Highest risks first**
- **Most educational or useful for development first**
- **Synchronise with other developments (e.g. hardware)**
- **Every cycle delivers a useful, *completed*, working result**

Delivery selection criteria

Juiciest, most important stakeholder values that can be made in the least time

- **Every delivery must have symmetrical stakeholder values (features, qualities),** otherwise the stakeholders get stuck
  - **Delete ↔ Add**
  - **Copy ↔ Paste**
- **Every new delivery must have clear extras,** otherwise the stakeholders won’t keep producing feedback
- **Every delivery delivers smallest clear increment,** to get the most rapid and most frequent feedback
- **If a delivery takes more than two weeks, it can usually be shortened:** try harder

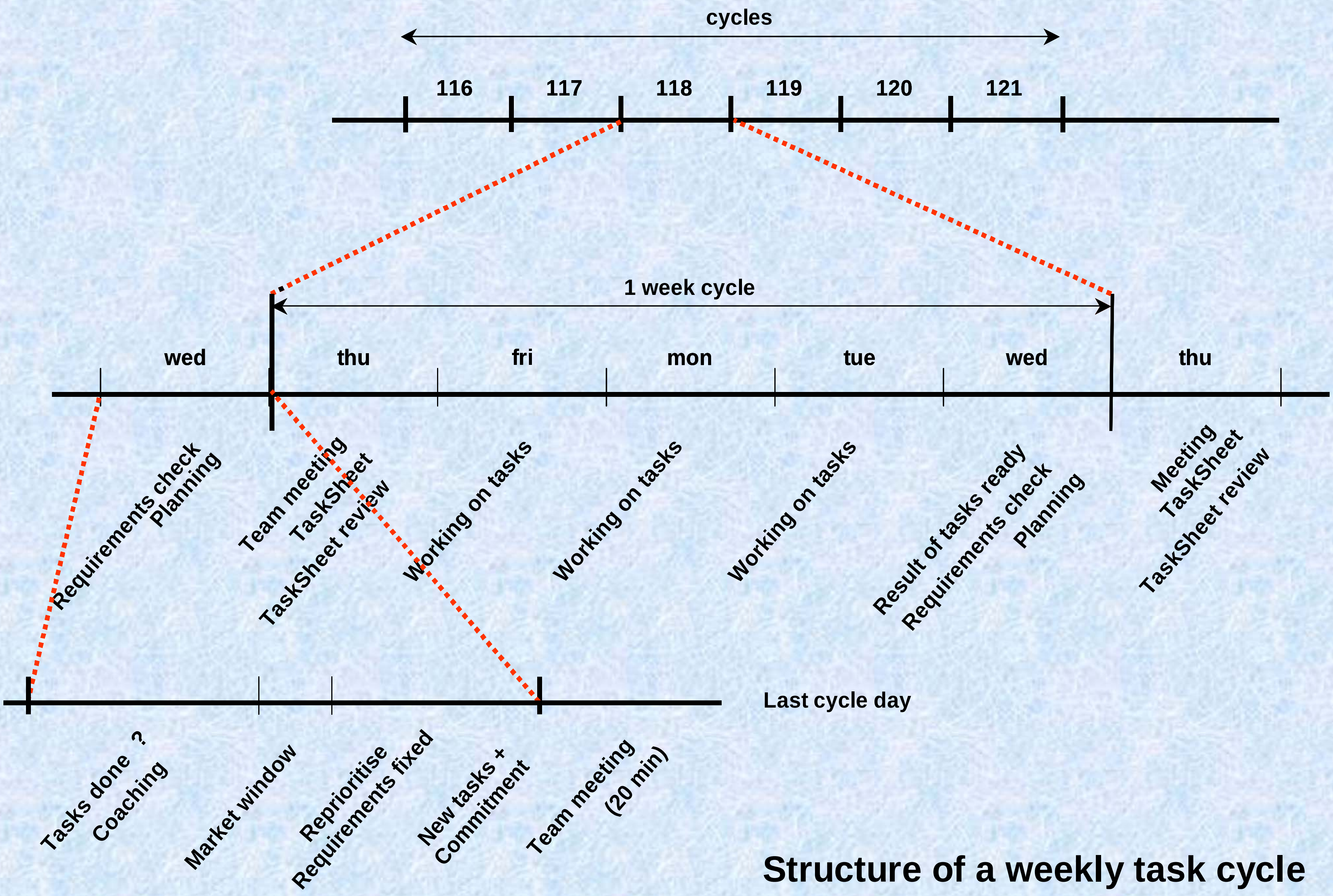
Dependencies



Priorities

Better 80% 100% done, than 100% 80% done

Let it be the most important 80%



cycles

116 117 118 119 120 121

1 week cycle

wed thu fri mon tue wed thu

Requirements check
Planning

Team meeting
TaskSheet
TaskSheet review

Working on tasks

Working on tasks

Working on tasks

Result of tasks ready
Requirements check
Planning

Meeting
TaskSheet
TaskSheet review

Last cycle day

Tasks done ?
Coaching

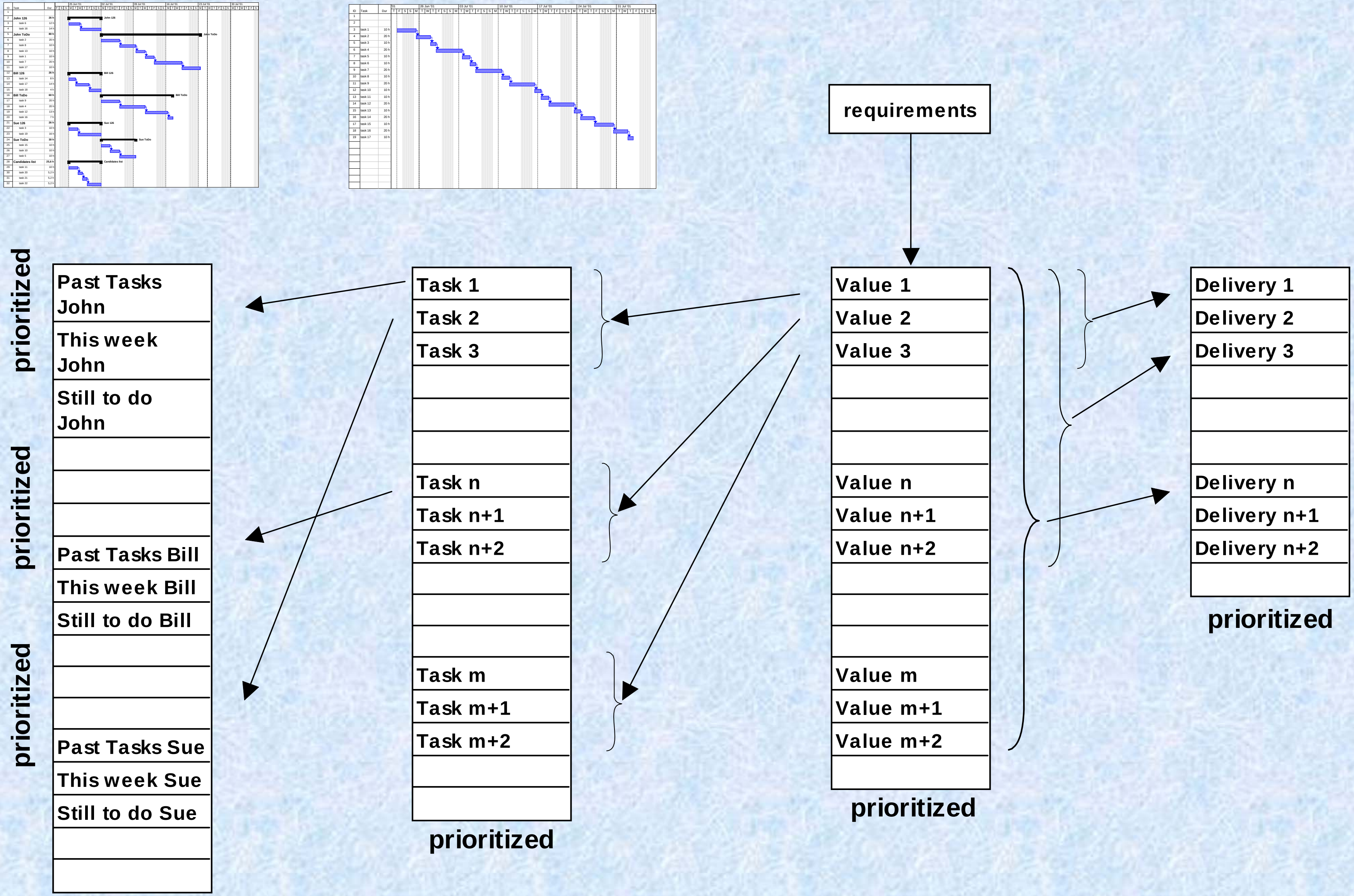
Market window

Reprioritise
Requirements fixed

New tasks +
Commitment

Team meeting
(20 min)

Structure of a weekly task cycle



requirements

Past Tasks John
This week John
Still to do John

Past Tasks Bill
This week Bill
Still to do Bill

Past Tasks Sue
This week Sue
Still to do Sue

prioritized
prioritized
prioritized

Task 1
Task 2
Task 3

Task n
Task n+1
Task n+2

Task m
Task m+1
Task m+2

prioritized

Value 1
Value 2
Value 3

Value n
Value n+1
Value n+2

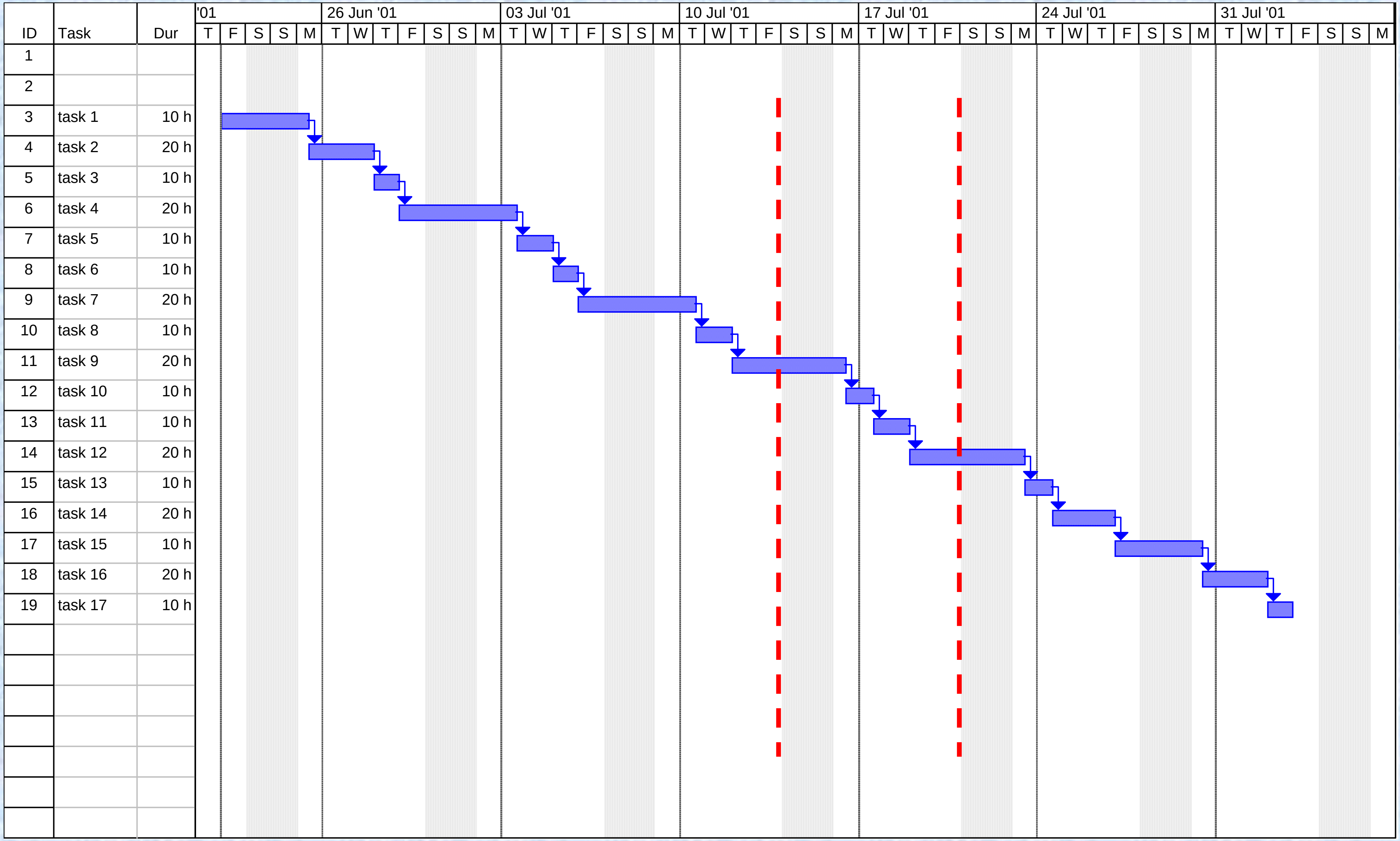
Value m
Value m+1
Value m+2

prioritized

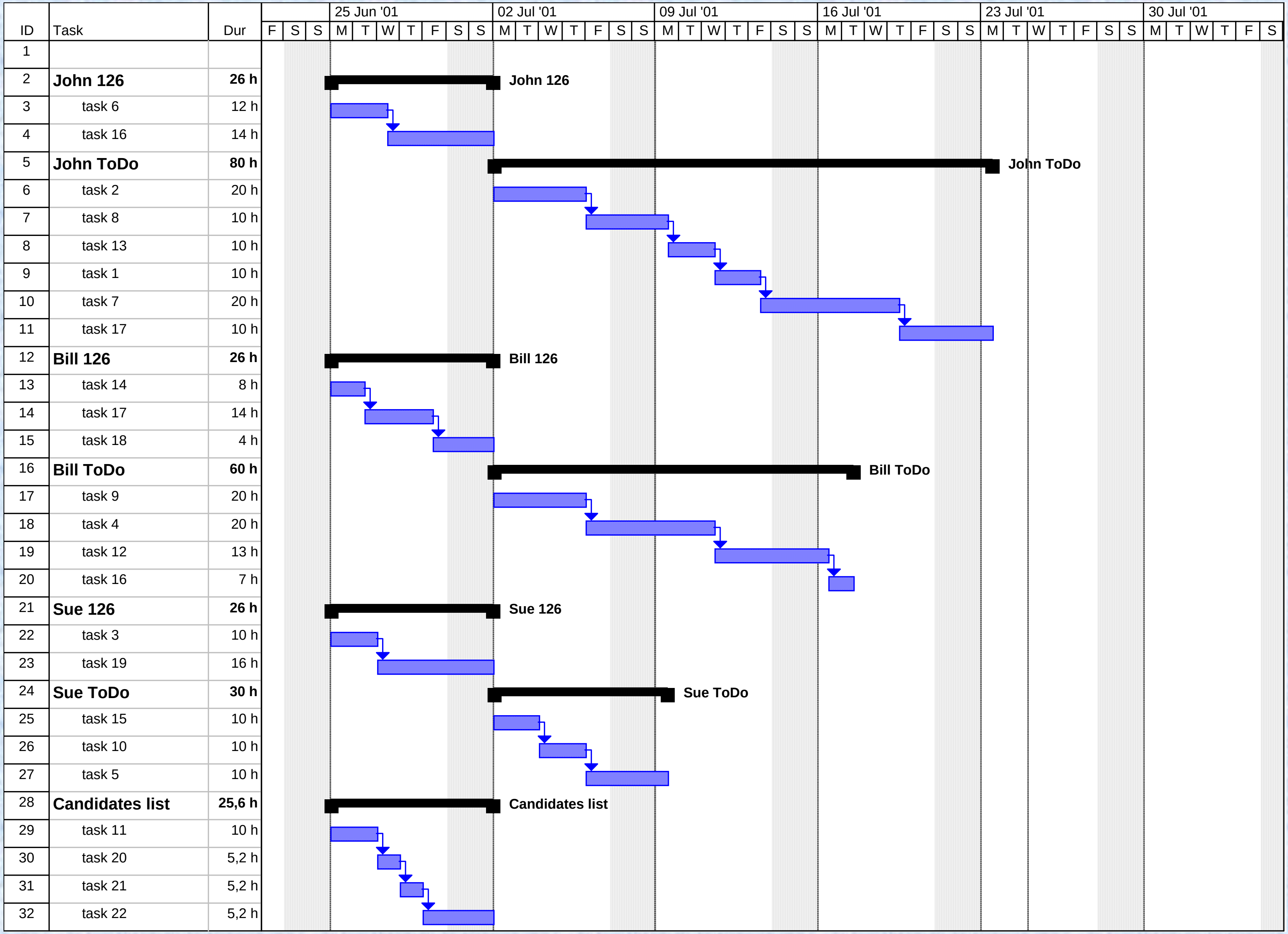
Delivery 1
Delivery 2
Delivery 3

Delivery n
Delivery n+1
Delivery n+2

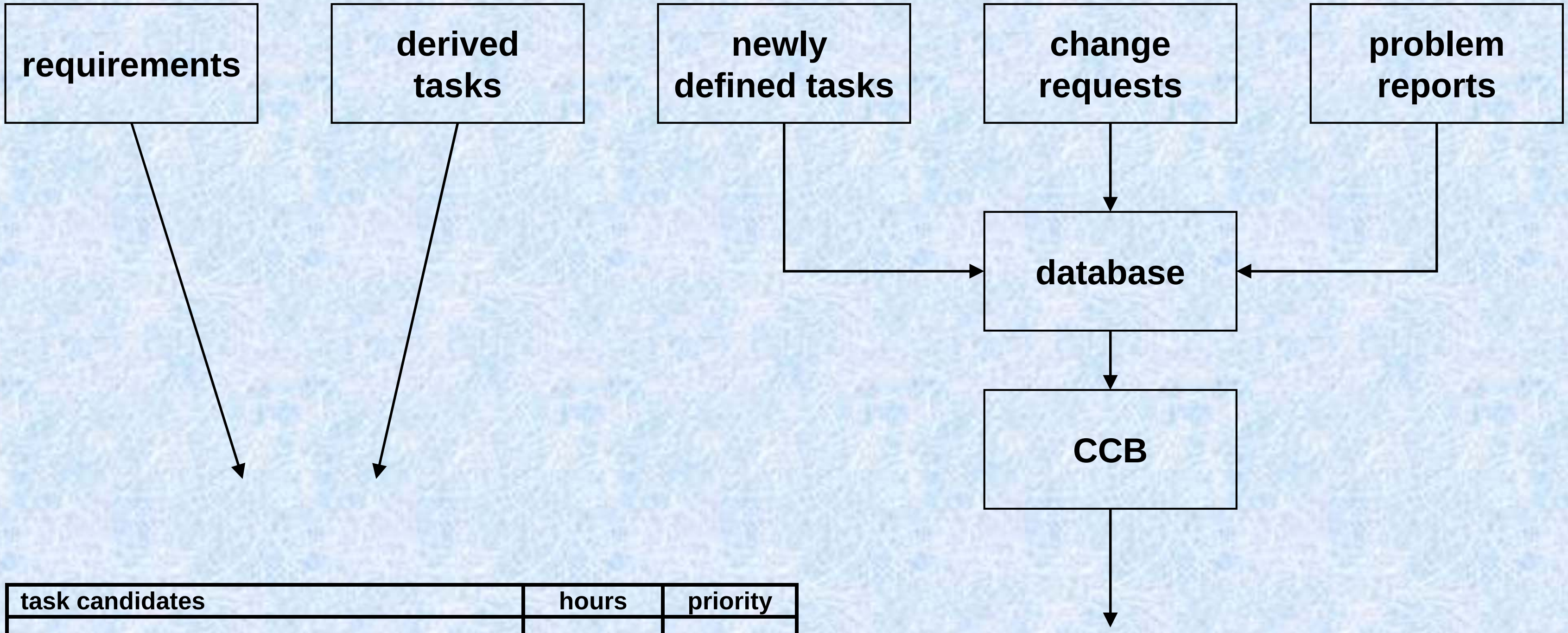
prioritized



ID	Task	Dur
1		
2		
3	task 1	10 h
4	task 2	20 h
5	task 3	10 h
6	task 4	20 h
7	task 5	10 h
8	task 6	10 h
9	task 7	20 h
10	task 8	10 h
11	task 9	20 h
12	task 10	10 h
13	task 11	10 h
14	task 12	20 h
15	task 13	10 h
16	task 14	20 h
17	task 15	10 h
18	task 16	20 h
19	task 17	10 h



ID	Task	Dur
1		
2	**John 126**	**26 h**
3	task 6	12 h
4	task 16	14 h
5	**John ToDo**	**80 h**
6	task 2	20 h
7	task 8	10 h
8	task 13	10 h
9	task 1	10 h
10	task 7	20 h
11	task 17	10 h
12	**Bill 126**	**26 h**
13	task 14	8 h
14	task 17	14 h
15	task 18	4 h
16	**Bill ToDo**	**60 h**
17	task 9	20 h
18	task 4	20 h
19	task 12	13 h
20	task 16	7 h
21	**Sue 126**	**26 h**
22	task 3	10 h
23	task 19	16 h
24	**Sue ToDo**	**30 h**
25	task 15	10 h
26	task 10	10 h
27	task 5	10 h
28	**Candidates list**	**25,6 h**
29	task 11	10 h
30	task 20	5,2 h
31	task 21	5,2 h
32	task 22	5,2 h



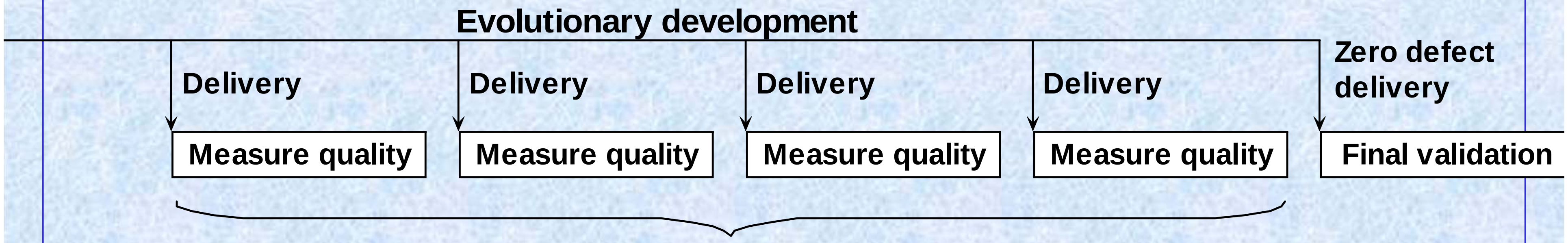
- Reject
- Later
- Analysis task
- New task

task candidates	hours	priority
task 1	12	5
task 2	22	5
task 3	13	5
|	17	4
|	|	4
|	|	3
|	|	2
|	|	2
|	|	1
|	|	0
task n	34	0

hours: real effort
priority: 5 = highest, 1 = lowest, 0 = on hold

Anything that must be done goes through the Candidate Task mechanism

Testing in Evo



how far are we from the goal of "zero defect delivery"?

- Final validation shouldn’t find any problems
- Earlier verifications mirror quality level to developers: how far from goal and what to learn

Magic words

- Focus
- Priority
- Synchronise
- Why
- Dates are sacred
- Done
- Bug, debug
- Discipline

Links

- **www.gilb.com**
  Evo guru
- **www.spipartners.nl**
  Simon’s website - Gilb’s courses in Holland
- **www.malotaux.nl/nrm**
  Niels’ website
- **www.malotaux.nl/nrm/Evo**
  Evo pages
- **www.malotaux.nl/nrm/pdf/MxEvo.pdf**
  Evo booklet

Can you afford *not* to use Evo?

Simon Porro
SPI Partners BV
+31- 40 - 248 98 22
porro@spipartners.nl
www.spipartners.nl

Niels Malotaux
N R Malotaux - Consultancy
+31- 30 - 228 88 68
niels@malotaux.nl
www.malotaux.nl/nrm/English