

SOFTWARE DEVELOPMENT PRACTICE

NUTS & BOLTS

Robert J. Schaaf

rjschaaf@ieee.org

September 12, 2019

© Robert J. Schaaf

SCOPE: ALL TYPES OF SOFTWARE

Applications

Business, science and engineering, **sense & control**, tools, middleware...

Configurations

Host-based, client-server, cloud-based, **embedded**, ...

Offerings

Custom, generic / off-the-shelf, component, **service**, ...

Flavors of software engineering

Development, operations, acquisition

©

What projects?

2

SCOPE: LARGE, COMPLEX PROJECTS

- Projects with more than **~10 people**
- Projects longer than **~6 months**
- **Long-life** software, assets – no throw-a-ways

“MORE IS DIFFERENT”

©

View from 30,000 feet above

3

VIEW FROM 30,000 FEET ABOVE

- Enormous **progress** over the past 50 years, particularly in **size**
- Shift from hardware-defined to **software-defined** systems, services
- Still, too many **failures** – in development, acquisition, operations
- In many cases, big **uncertainties** in time, cost, quality and risk
- Software engineering is **situational** – checklists don't work
- Still missing: Framework for **general-management control**
- Real: **DO'S & DON'TS** for given projects – “Nuts & Bolts”

©

What subjects?

4

TOPICS FOR THIS EVENING

- Requirements
- Project Management



Requirements

5

REQUIREMENTS

+

- Stakeholders
- Stakeholder Needs
- Consensus, Quality



Requirements Practice

6

REQUIREMENTS PRACTICE

- The requirements practice remains **controversial**
- “We have no time for requirements,” “Requirements always change,” etc.
- Alternatives – e.g. project charter, stories – not enough for complex projects
- **Often, we have to fight for the practice of requirements**
 - In effect, requirements set the scope of a project
 - Yes, requirements do change, and that’s OK, we can handle it

©

Stakeholders

7

STAKEHOLDERS

Cast the net **much wider than *customer and user***

A stakeholder is a person with an interest in anything related to the software:

- A. The software itself, incl. **any property or attribute** of the software
E.g.: the customer(s); the owner(s); the users
But also, say the legal department
- B. The **environment** in which the software will be used
E.g.: the owner of a system interacting with the software
Or, say the designer of the application’s business processes
- C. The **processes** applied during the life of the software
E.g.: developers; acquirers; trainers; sub-contractors
But also: internal audit, finance, marketers, sales, pricing, etc.

Trade-off

©

Stakeholder Needs

8

Distinguish between *stakeholder needs* and *software requirements*

STAKEHOLDER NEEDS

- **Stakeholder need:** A condition that must or should be resolved or fulfilled according to one or more stakeholders
 - Strength of need **varies** – what value to stakeholder?
 - A need is owned/held by the respective stakeholder(s)
 - Others *may* help in the discovery & formulation of needs
- Satisfying a stakeholder need may depend upon
 - The software **itself**
 - The interoperation between **environment** and software
 - The **processes** to engineer the software
- **More than Function:** Endless variety of types of needs

©

Sample Types of Stakeholder Needs

9

SAMPLE TYPES OF STAKEHOLDER NEEDS

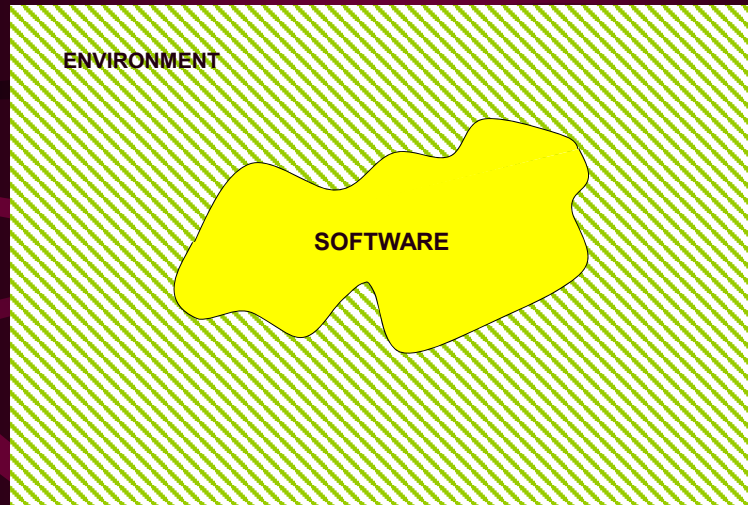
Functionality	Completeness	Environmental factors
Schedule	Maintainability	Usability, human factors
Timeliness	Configurability	Operability
Cost, price	Flexibility	Installability
Quality	Scalability	Security
Accuracy	Privacy	Modes of operation
Performance	Adaptability	Privacy
Capacity	Portability	Integrity
Responsiveness	Interoperability	Property rights
Efficiency	Extendibility	Merchantability
Effectiveness	Compatibility	Ethical values
Safety	Simplicity	Regulatory compliance
Reliability	Modularity	Standards compliance
Availability	Reusability	International factors
Serviceability	Testability	Auditability
Robustness	Trainability	<u>AND SO ON</u>

©

The Subject Software in Its Environment

10

THE SUBJECT SOFTWARE IN ITS ENVIRONMENT

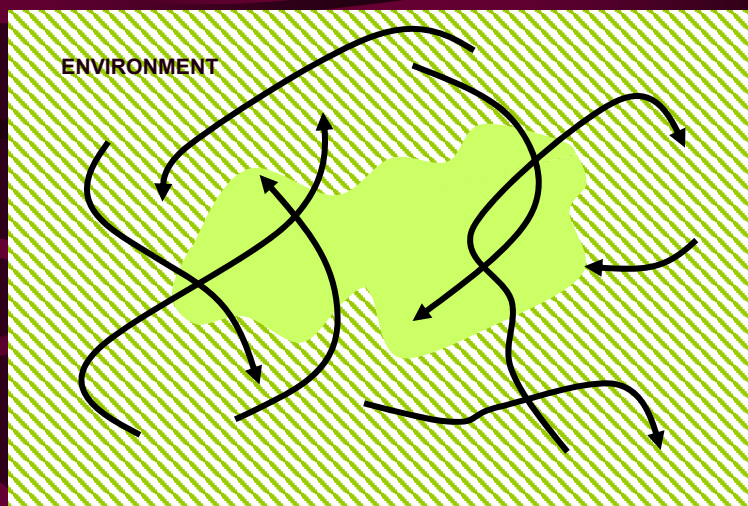


Domain

Stakeholder Needs

11

STAKEHOLDER NEEDS



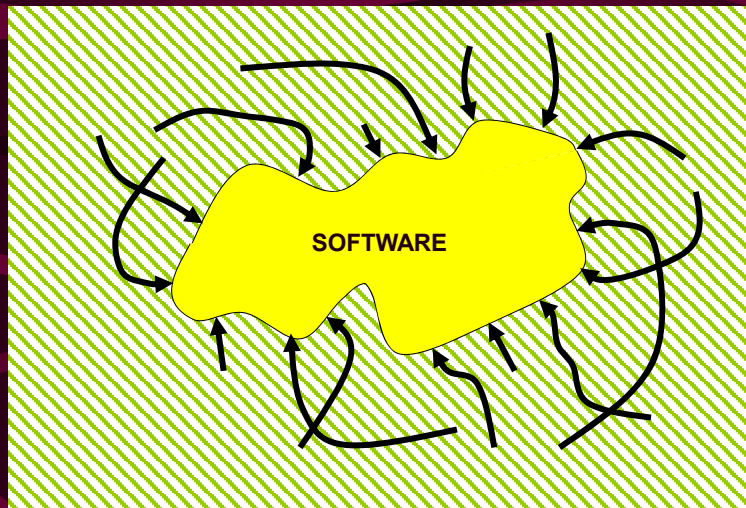
Needs should be **irrespective** of the subject software's boundary

Domain problems, domain goals

Software
Requirements

12

SOFTWARE REQUIREMENTS



Software requirements **delineate** the subject software

©

What definition? 13

SOFTWARE REQUIREMENT

"A condition on the software, or on the engineering process, to make the software acceptable to the stakeholders"

- Requirements vary in strength: "must," "should", "may", etc. – all requirements are not equal
- Requirements should be stated in the positive – negatives are hard to prove
- Requirements, like needs, may change

"There should be no security holes"

©

Requirement Types

14

REQUIREMENT TYPES

While needs address the stakeholders' domain of interest.....

Requirements must address everything that may influence the acceptability of the software

For example, requirements addressing: function; performance; reliability; availability; safety, security; installability; compatibility; scalability; **software processes; schedule**; etc.

©

Who, What, How and When

15

WHO, WHAT, HOW AND WHEN

STAKEHOLDER NEEDS	SOFTWARE REQUIREMENTS
Responsibility of stakeholders	Responsibility of project manager
What the problem is	What the solution should be
Scope: Domain	Scope: Software
Business processes, workflows Domain scenarios, stories	Use cases, test cases Finite state machines, etc.
Mostly reactive: Feedback	Mostly pro-active
Throughout the project	Throughout the project

©

Consensus and Quality

16



CONSENSUS AND QUALITY

- For **stakeholder consensus**, use requirements as basis on which to reach consensus – not needs, they may have internal conflicts
 - **Consensus during the project** about what is to be developed
 - **At the end of the project** that the actual software is acceptable
- Transformation from needs to requirements is **non-deterministic**, same needs may lead to different requirements – opportunity!
- A requirement must be formulated in a way that makes it possible to test for conformance **in the course of the project**
- Working definition of quality: “The degree that the software, at the end of the project, **meets software requirements**”

©

Q1: What If Disagreement on Requirements?

17

Q1: WHAT IF DISAGREEMENT ON REQUIREMENTS?

- Project manager **negotiates**, in terms of software requirements, *not* stakeholder needs
- **Consult & Cajole – NO Command & Control**
- **Focus on bottom-line: Profit, mission**
- **Persistent disagreement? Case: IBM's Future System**

©

Q2: What If Requirements are Defective?

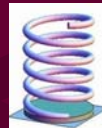
18

Q2: WHAT IF AGREED REQUIREMENTS ARE DEFECTIVE?

Worse, requirements may be unknowable, at least initially

Several root causes of unknowability

The more software, the more unknowability



ANSWER

Incremental evolution of needs > requirements > software

DEFINE QUALITY AS

'SATISFYING THE STAKEHOLDERS'

TREAT 'MEETING REQUIREMENTS' AS A PROXY, STAND-IN

©

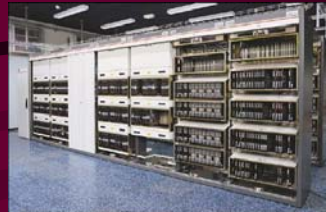
Software Project management 19

SOFTWARE PROJECT MANAGEMENT

©

Case: System X 20

CASE: SYSTEM X



- **Large, complex** communications software system
- Embedded in a new, **make-or-break** product family, “X”
- 1000 software engineers, 10 labs, 3 continents
- Approach: Develop for lead customers, then spread out
- Continuous coordination problems, mostly **B <> G**
- Enter the **first** software project manager.....

©

Geography

21

GEOGRAPHY



CORPORATE GOALS IN THE BACKGROUND:
**GLOBAL MARKET
CONFEDERATION > FEDERATION**

Typical sources of project failure 22

TYPICAL SOURCES OF PROJECT FAILURE

- #1 Project **coordination** instead of project management
- #2 Invisible stakeholders, no requirements
- #3 Little or **no planning**, or inflexible
- #4 Project plan **implausible** vis-à-vis past performance
- #5 Poor **control over execution**
- #6 Unmanaged **risks**, small problems left to grow
- #7 Unreasonable **expectations** go unchecked
- #... **Staffing**

Familiar problems, known solutions



PROJECT MANAGEMENT RESPONSIBILITIES

- Following 3 slides show the **customary** project management responsibilities
- Distinguish between “project **management**” and “project **manager**”
- **Good practice**: Within the scope of anybody’s work, and with a few noted exceptions, **all staff** share in the project management responsibilities



PROJECT MANAGEMENT RESPONSIBILITIES – Cont'd

1. Organization – The Who

- Decide Project vs. Functional organization
- Assign personnel, responsibilities, accountabilities, authorities
- Set rules of governance, delegation

2. Project Scope – The What + How

- Identify stakeholders, elicit stakeholder needs, negotiate requirements
- Break down of product, work & output
- Maintain link between {needs, requirements} and {work, output}

3. Use of Time – The When

- Sequence activities, find dependencies and assumptions
- Estimate resources x time for each activity
- Create & maintain project schedule, verify against resource planning etc.

©

Project Management Responsibilities 25

PROJECT MANAGEMENT RESPONSIBILITIES – Cont'd

4. Cost

- For each activity, estimate the costs of the resources needed
- Aggregate the estimated costs in a budget, suitable for control
- Control actuals against budget, control changes in budget

5. Quality

- Establish and maintain a quality management system
- Ensure, measure and control product and process quality
- Improve processes and tools to meet requirements

6. Staff

- Develop and execute a staffing plan
- Analyze actual competencies against needs, plan training if necessary
- Track individual and team performance, provide feedback, resolve issues

©

Project Management Responsibilities 26

PROJECT MANAGEMENT RESPONSIBILITIES – Cont'd

7. Communication

- Determine information needs > plan the flows > set the tone
- Motivate project personnel – recognize, explain
- **Massage stakeholder expectations**

8. Risk

- Establish a **risk management system**, control its performance
- Track major dependencies and assumptions, **analyze plan deviations**
- Identify risks - continuously
- Mitigate risks: alternative courses of action, fallback plans

9. Procurement

- Set make-or-buy direction, approve make-or-buy decisions
- Identification and selection of sellers; contracting + fulfillment

©

A project is....

27

A project is....

- A way of organizing work
- Work with a beginning and an end
- Work that leads to a unique result
- As opposed to: Ongoing, repetitive work

©

Project and Organization

28

PROJECT AND ORGANIZATION

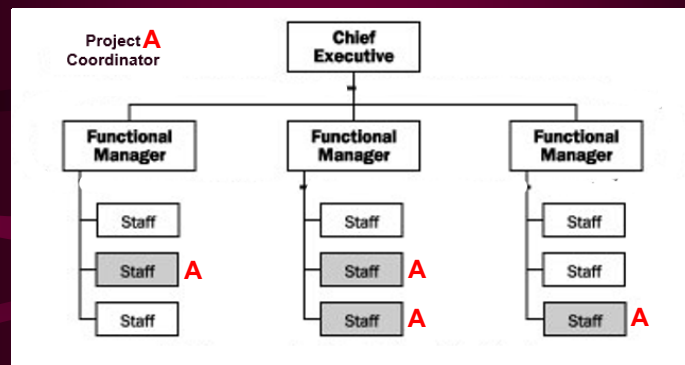
- Functional organization
- Project organization
- Mixed organization

©

Functional Organization

29

FUNCTIONAL ORGANIZATION – WORK TO THE PEOPLE



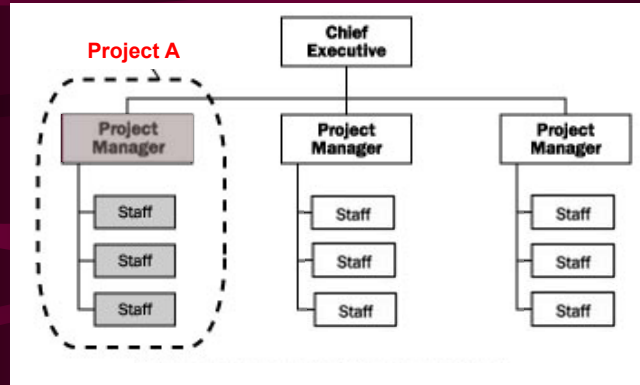
- People understand their own work, but not the total
- Friction in pushing projects through, sub-optimization
- Unfriendly to complexity and uncertainty
- Works for software maintenance and operations

©

Project Organization

30

PROJECT ORGANIZATION – PEOPLE TO THE WORK



- Organize in small teams (10-15) + fully-responsible project manager
- With managers reporting to project manager, scales up to ~200 staff
- Flexibility, accommodates complexity and uncertainty
- People understand the total – not so much their own contribution
- Nowadays, dominant model for software development

©

System X Software Organization

31

SYSTEM X SOFTWARE ORGANIZATION

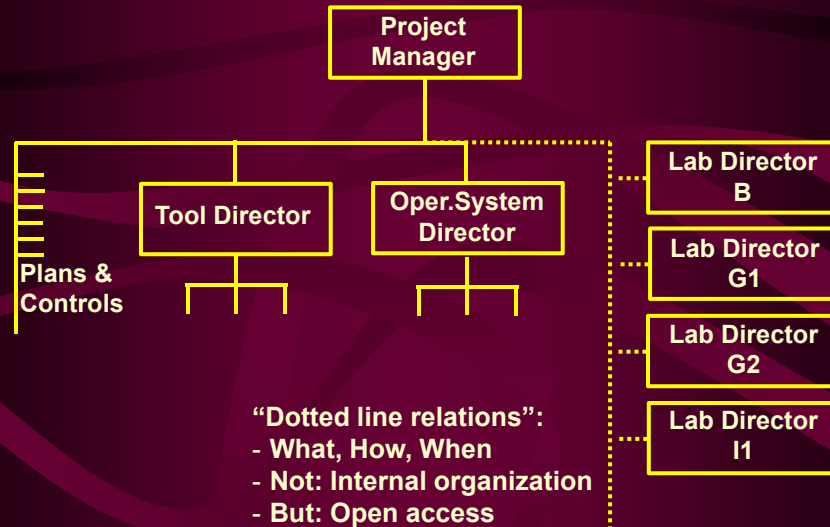
- Large, complex communications software system
- 1000 software engineers, 10 development centers ("labs"), 3 continents
- Enter the first project manager....
- Full project management **responsibility + accountability**
- Direct authority over **1/3** of staff
- Controlling **operating system, tools, plans & controls**
- 'Dotted line' relationship to 8 of 10 Lab Directors

©

Organization Chart

32

SYSTEM X SOFTWARE ORGANIZATION



My Priorities 33

MY PRIORITIES AS PROJECT MANAGER

CUSTOMERS, NEEDS
PLAUSIBLE PLAN
CONTROL OVER EXECUTION
RISK
PERSONNEL

Hidden Problems 34

HIDDEN SYSTEM X PROBLEMS

- No access to lead customers
- Main product feature (distributed control) not yet proven
- Emergent properties > Maintaining intellectual control
- Trading off among system properties
- Distributed development w/o autonomous sub-projects
- Conflict between “phases” and “agility”

**SYSTEM X WAS A BET-THE-COMPANY ENDEAVOR
WITH FLAVORS OF A SCIENCE PROJECT**



Customers, Needs

35

CUSTOMERS, NEEDS

- Often, sales people protective of their customers
- System X: 2 lead customers, in B and G
- Links between these customers and the local labs
- Except these labs, nobody else met these customers
- Needs? What had they been told? What did they think?

With System X: Project Manager left ‘in the dark’



How Do We Get a Plausible Plan?

36

HOW DO WE GET A PLAUSIBLE PLAN?

Three possible starting points:

- Breakdown the **work** > Amount of work
- Breakdown the **product** > Size of product
- Use history: “Done this before”

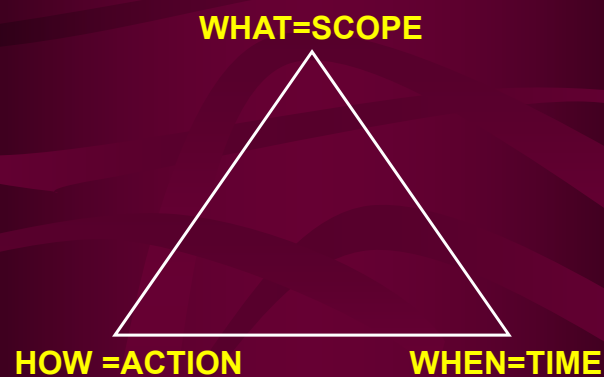
Our favored method: **Planning based on product size**

- Unit of measure: New/Changed Source Statement “Loc”
- Plan activities **incrementally**, as time progresses
- Plan close-in activities in **detail**
- **Rough** outline for the activities further out

©

“Golden Triangle” of Project Planning 37

‘GOLDEN TRIANGLE’ OF PROJECT PLANNING



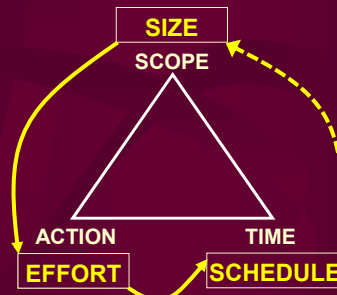
©

Size-Based Planning

38

SIZE-BASED PLANNING

- From requirements (> design specs), estimate **product size**
- From size, estimate **effort** – for precision, also consider complexity etc.
- Based on size and effort, set (adjust) staffing level and time **schedule**
- If necessary, **re-negotiate** requirements to affect size > effort > schedule

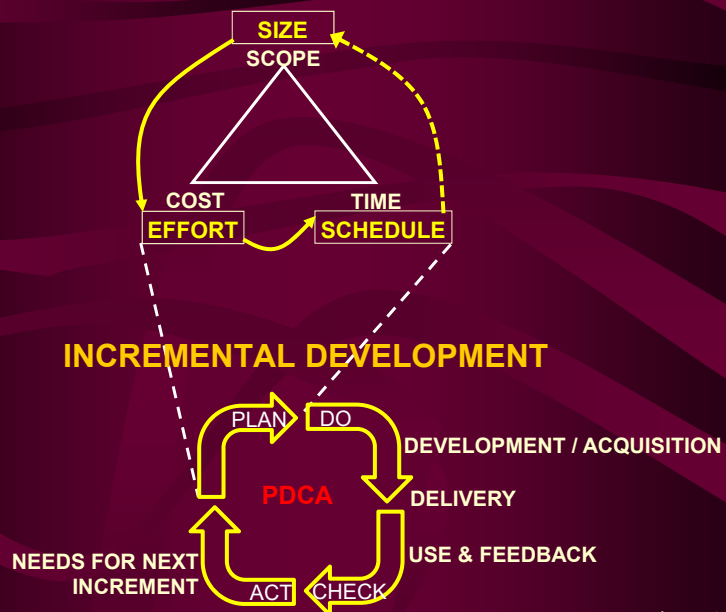


ITERATIVE PROCESS!

Incremental Development

39

ITERATIVE PLANNING FOR EACH INCREMENT



Increments



INCREMENTS

- Planning is predicting the future
- Predicting the future is difficult
- How far out can we plan with reasonable confidence?
- My rule-of-thumb for software development:

Team size

10

40 (in 3-5 groups)

200+ (hierarchy of 2-3)

Planning horizon = Increment

~1 month

~3 months

~6 months



- For instance, in System X:

Tool Lab, 60 engineers

New planning every quarter

Each of 4 tool groups

New planning every month

System X Software Project

New planning every 6 mo.s

- No great need to sync the various planning activities

©

Forecasting. 41

FORECASTING

- "Plans go out the window when the first bullet is fired."
- Planning informs the decision what to do next
- Planning is continuous, there's always 'and next?'
- Continuously moving the goal posts?
- Committed ("the plan") and forecasted completion dates
- Example: Design XYZ – P: Sep 20, 2019; F: Sep 27, 2019
- Only the planning of next increment resets commitments

©

A Plausible Plan Also Has.... 42

A PLAUSIBLE PLAN ALSO HAS....

Commitment, buy-in – explicit, from all actors, and lasting

- Including buy-in + approval by the direct manager
- No commitment, buy-in or approval means no plan
- Review, comments and silence are no commitments

Brevity - No unnecessary detail, no unnecessary precision

- Only specifics that will be checked, measured, counted
- Only specifics where deviation would require extra action
- No guidelines, no project introduction – plan is not for training

Common sense – E.g., no plan without contingency resources

©

Control over Project Execution

43

CONTROL OVER PROJECT EXECUTION

- Control with a light touch
- To some degree, “Let things happen”
- System X: **Plans & Controls** < 25 staff
 - Plans: Are plans aligned?
 - Design Control: Are requirements whole?
 - Integration & Test Control: Quality assured?
 - Find disconnects, then lead **negotiation**
- Reliance on **managers doing their jobs**
- Framework of **reports and meetings**

©

Managers Managing to Every Plan Item

44

MANAGERS MANAGING TO EVERY PLAN ITEM

- Corollaries: **That which is not managed will not happen except by luck**
- If it's "not that important," it should not be in the plan.
- Impossible to achieve every plan item – **recognize deviation** and recover, locally or by plan changes: cutbacks, delays, additional resources....

©

Deviations May Come in Many Forms 45

DEVIATIONS COME IN MANY FORMS

Examples

- Activities not finished in time
- Outputs not available in time
- Dependencies not satisfied in time

BUT MANY OTHER MEASURABLE/OBSERVABLE DEVIATIONS COME EARLIER IN TIME:

- Sizes larger than planned
- Staffing shortfalls
- Complexity higher than planned
- Hidden assumptions, hidden dependencies
- Appearance of important outputs not shown in the plan
- Substantial activities underway not shown in the plan
- Activities taking more staff than planned
- More re-work than planned

©

Framework of Reports, Meetings 46

REPORTS, MEETINGS, DECISION-MAKING

- At the **individual** level: Daily face-to-face meetings
- At manager's levels: Weekly reports of
 - Commitments met
 - Red Flags raised + resolution plans
 - Forecast changes
- Monthly meetings Project Manager with Lab Directors
 - Review, discuss, explain
 - Directions, decisions – **Careful with decisions!**

©

Reporting by Project Manager 47

REPORTING BY THE PROJECT MANAGER

- Project manager's **Progress & Issues** report, weekly
- **Regular reviews**, typically once per month – System X: At HQ
 - Project manager is lead presenter – often the only one
 - Basically for information purposes – System X: For HQ staff
 - Good questions + No, or minor, course corrections
- **Major gates**, typically with six-months intervals
 - Verification against the last six-months detailed plan
 - Stakeholders provide their assessments
 - Project manager presents next six-months plan
 - Go/no-go decisions: **Incremental commitment**

©

Risk

48

RISK

- Risk handling should be **continuously & distributed**
- **Broken Windows Policy**: Tackle small deviations
- Find and re-work **error-prone** modules
- **Root cause analysis**, process assessment
- **Measurements** for quantification and pre-emption
- Manage **expectations...**

©

Manage Expectations 49

MANAGE EXPECTATIONS

Forces for good and bad – motivational and destructive

Communicate – Up, down, side-ways

Plain-speaking, even bluntness, is good

Generally, audiences have a **great sense of reality**

©

Staffing 50

STAFFING

- Numbers don't make up for too few good people
- Avoid bubbles or steps in staffing level
- Easiest: Maintain level headcount
- Look for a good number of "generalists"
- **Implement** staffing plan with sense of urgency
- "Making-up later" for shortfalls near-impossible
 - Extend schedule, or
 - Cut back on scope

©

Pick the Right People to Work With 51

PICK THE RIGHT PEOPLE

People's performance follows an exponential curve:

- Good people "10x" better than average people
- Average people "infinitely" better than poor people

Poor performers make work for other people

Average people *do* work

Good people **solve problems, minimize the work**

What to do with poor performers? Identify them, then

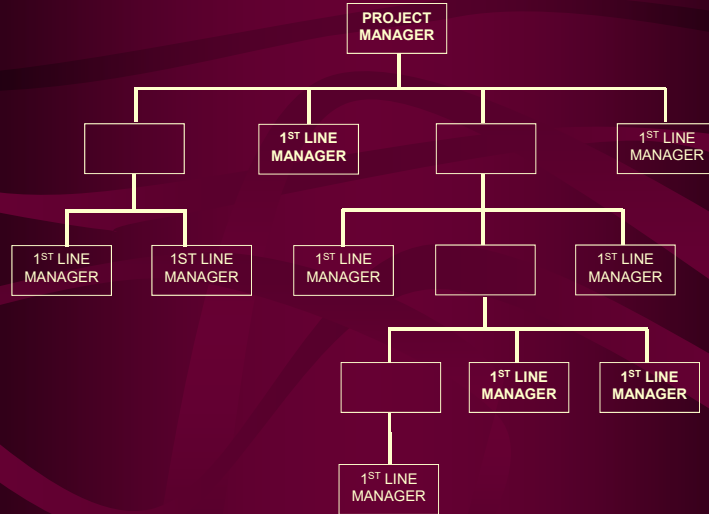
- Grow them
- Or move them out

©

First Line Managers

52

FIRST LINE MANAGERS



©

First Line Managers

53

FIRST LINE MANAGERS

TECHNICAL AND PERSONNEL LEADERS

Technically excellent, able to lead people

Able to take over from any person in group, break in new people

Principal contributor to the design of the software

Responsible for reviewing **100% of the code** of his/her group

Responsible for group-level planning, tracking and achievement

Responsible for work environment, morale, mentoring of individuals

Size of group needs to be limited to make the above realistic

Manager gets to learn who the good/average/poor people are

Manager finds out early whether the software will work

Manager can step in in case of staffing emergencies

Manager's review better and cheaper than code inspections

Benefits justify the increased number of managers required

©

The System X Story

54

THE SYSTEM X STORY

- Release 2
 - Mainly by B lab*, for local lead customer
 - **Good design**, chaotic process, limited function
- Release 3
 - Mainly by G labs*, for local lead customer
 - So-so design, **good process**, limited function

* But using the unified operating system and tools

©

Continuous Integration

55

CONTINUOUS INTEGRATION

As opposed to 2- or 3-week *'agile sprints'*

- Overnight, every night
- Compile + Build of everything checked-in
- Regression Test of everything that worked before
- If Regression Test fails, fall back to previous night's build

Always a working base for check-ins
No regression from what previously worked

©

The System X Story Continues

56

THE SYSTEM X STORY CONTINUES

- Release 2 – B: **Good design**, chaotic process, limited function
- Release 3 – G: Lesser design, **good process**, limited function
- ~~Release 4 – Target for convergence, full function~~
Still no access to B and G lead customers
Country Mgrs (stakeholders) disapprove R4
Release 4 abandoned
Lesson: You can't be your customer

©

My project manager life saved

57

MY PROJECT MANAGER LIFE SAVED

- A new customer, N !
- HQ: N to serve as lead customer for Release 5
- HQ: **B and G** to deliver to N
- Effectively, B and G had to drop their separate ways
- **Direct access to customer N for needs and feedback !**

©

How Did the System X Story End...

58

HOW DID THE SYSTEM X STORY END.....

- Release 2 – B: Good design, chaotic process
- Release 3 – G: Lesser design, good process
- ~~Release 4 – Target for convergence, full function~~
- **Release 5 – Unified design, continuous integration**

AFTER ALL

System X became a world-wide commercial success

Served by a single software system > Profit margin!

Series of Release 5 increments, never a Release 6

©

Q: What applies to smaller projects? 59

Q: WHAT APPLIES TO SMALLER PROJECTS?

- Project manager + Good people
- Stakeholders + Needs + Negotiated requirements
- Size-based planning (or stories/sprint?)
- Continuous integration
- Quality / measurement culture
- **SHORTER:** Planning horizons, Delivery times
- **MORE:** Acquisition
- **MORE:** Pair programming
- **LESS:** Reporting + Process

©

Ethical Value Engineering 60

ETHICAL VALUE ENGINEERING

- **Triggers: Artificial Intelligence, Autonomous Systems**

Example: Medical Referral System

Sample values: Trustworthy, Intelligent, Unbiased, Privacy, Honesty, Comfort, Patience, Empathy, Transparency, Dignity, Freedom, Autonomy, Respect, Integrity

- **How to get what ethical values into a target system**

IEEE, ISO initiatives: Development process guidelines

- **Two main alternatives for the guidelines:**

1. Ethical value engineering as **appendix** – what values
2. **Integrated** in standard development process – what + how

©

The Take-Away Message

61

THE TAKE-AWAY MESSAGE....



©

62

- Quality is the catalyst in Software Development
- Stakeholder satisfaction + Staff motivation + Less rework
 - Rework slows down a project disproportionately
 - Meanwhile, the project cost meter keeps running
- With less rework, the cost savings of a shorter schedule
erase any “cost of quality”

BOTTOM LINE: QUALITY IS FREE