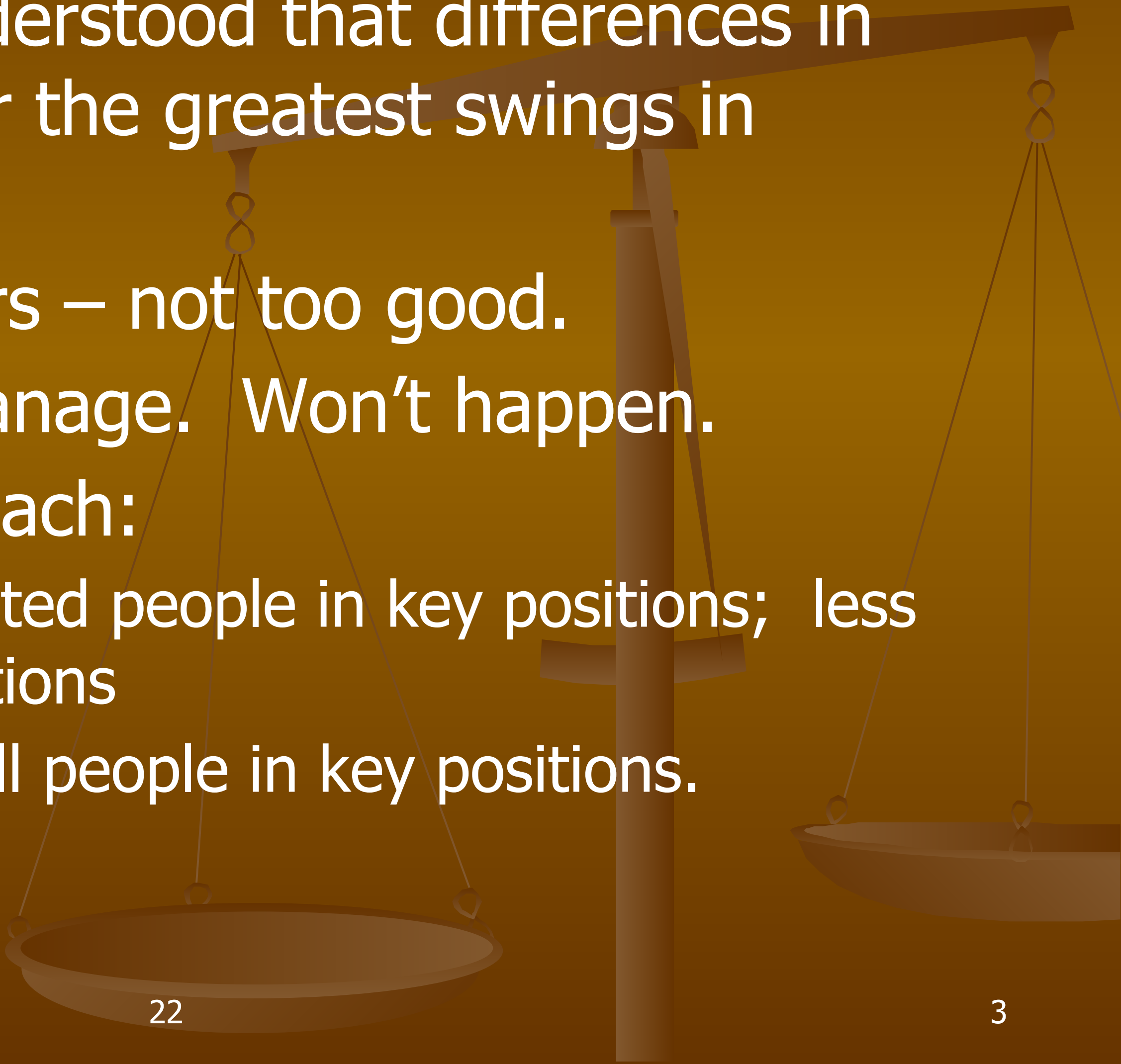
Chapter 3 – Improving Software Economics

Part 2 of 2

Outline

- 3. Improving Team Effectiveness (5)
- 4. Improving Automation through Software Economics (3)
- 5. Achieving Required Quality (5)
- 6. Peer Inspections: A Pragmatic View (7)



3. Improving Team Effectiveness

- "It has long been understood that differences in personnel account for the greatest swings in productivity."
- Great teams – all stars – not too good.
- Also impossible to manage. Won't happen.
- Best pragmatic approach:
  - Balance – highly talented people in key positions; less talented in other positions
  - Coverage – strong skill people in key positions.

3. Improving Team Effectiveness - continued

- Managing the team is the key.
- A well-managed team can make up for other shortcomings.
- Boehm’s recommendations:
  - 1. ☐ Principle of top talent: use better and fewer people.
    - Proper number of people is critical.
  - 2. ☐ Principle of job matching (skills and motivations)
    - Not uncommon in development teams for individuals to have a vision of promotion from programmer to project manager or to architect or to designer…

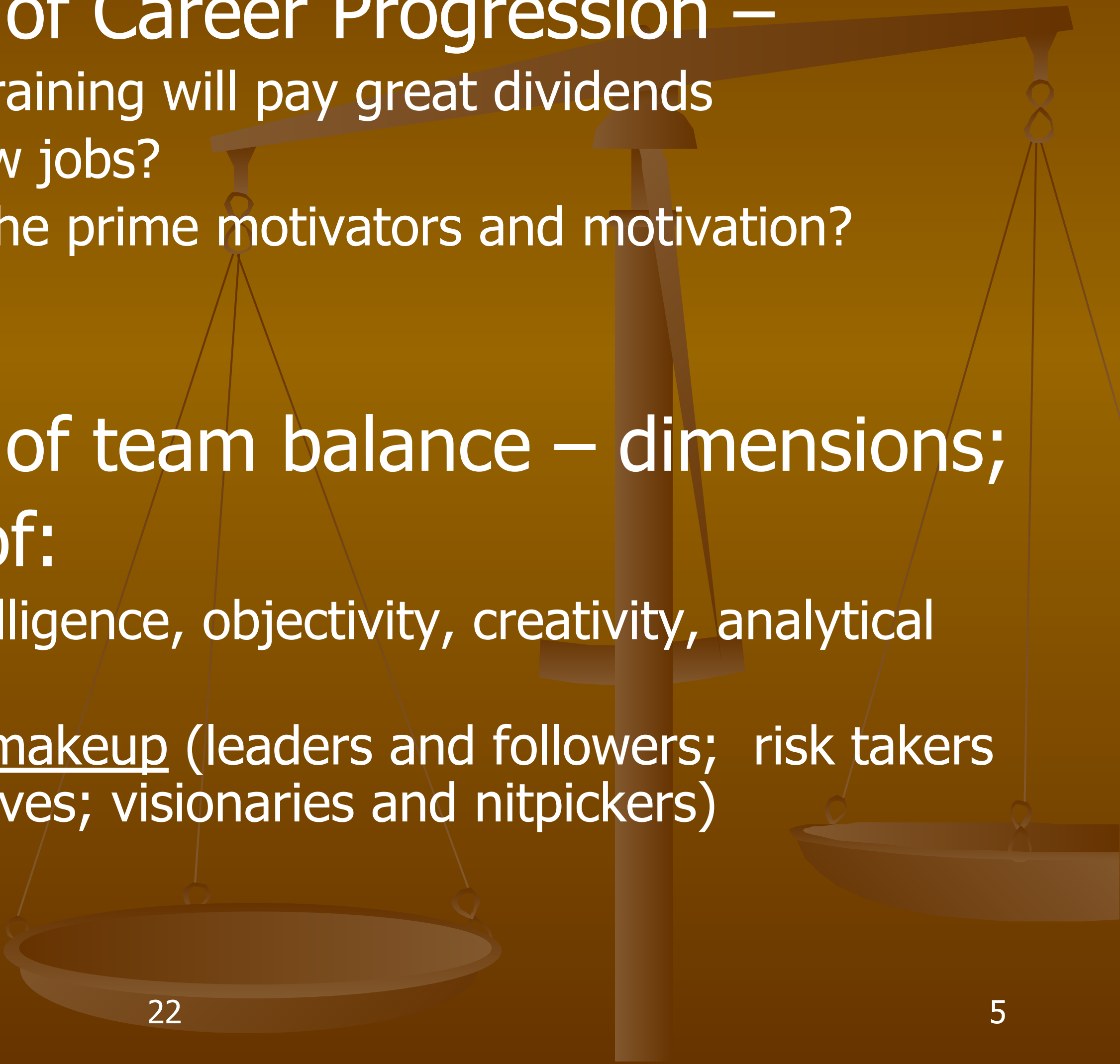
    - Skill sets are NOT the same and many projects have gone amuck due to poor management!

    - Great programmers are not necessarily great managers and conversely.

3. Improving Team Effectiveness - continued

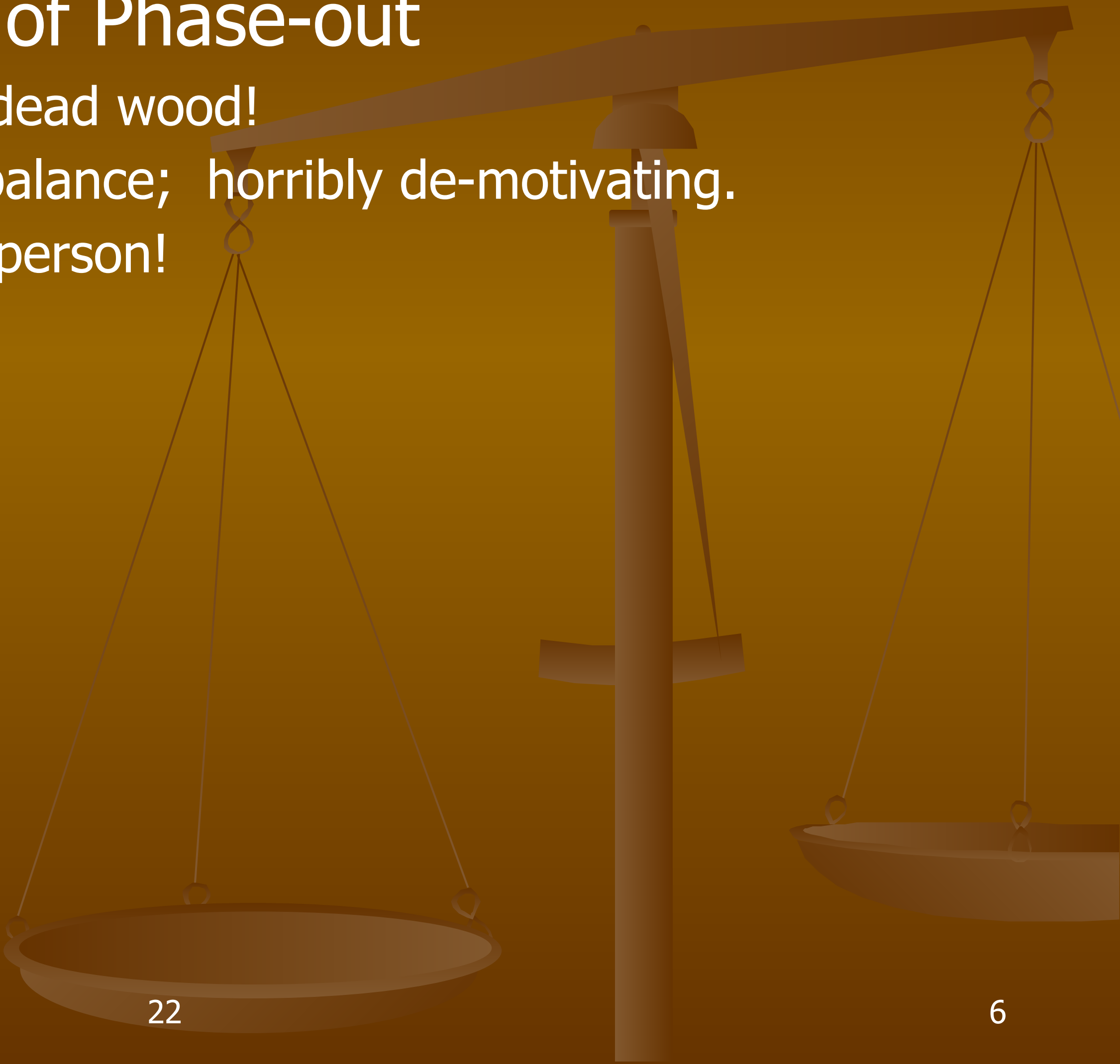
- 3. □ Principle of Career Progression –
  - Organization training will pay great dividends
  - Posting for new jobs?
  - □ What are the prime motivators and motivation?

- 4. □ Principle of team balance – dimensions; balance of:
  - <u>raw skills</u> (intelligence, objectivity, creativity, analytical thinking…)
  - <u>psychological makeup</u> (leaders and followers; risk takers and conservatives; visionaries and nitpickers)

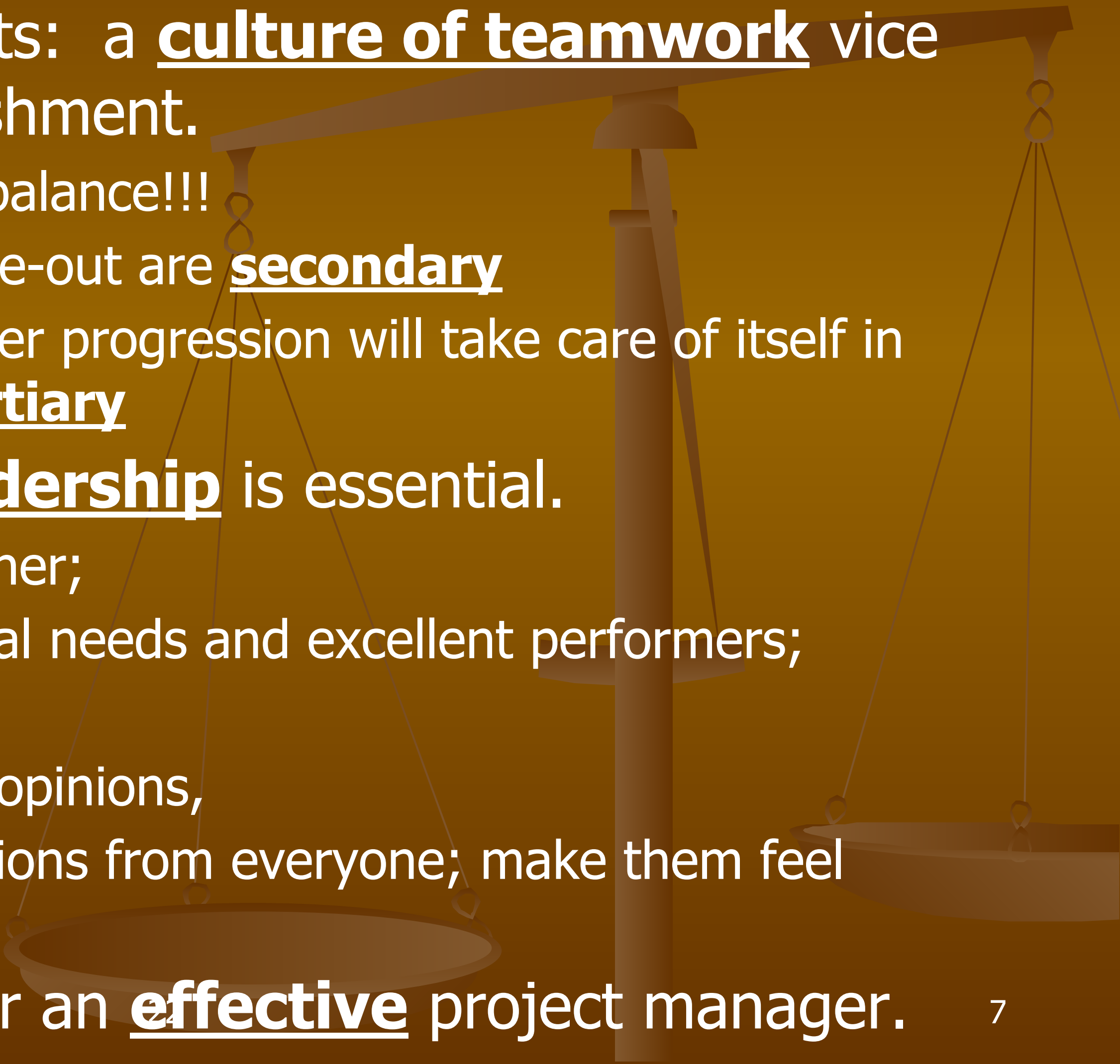


3. Improving Team Effectiveness - continued

- ☐ 5. Principle of Phase-out
  - Get rid of the dead wood!
  - Disrupt team balance; horribly de-motivating.
  - Get rid of this person!



3. Improving Team Effectiveness – continued (last)



- Overall team guidance:
  - Essential ingredients: a **culture of teamwork** vice individual accomplishment.
    - ☐ Teamwork and balance!!!
    - Top talent and phase-out are **secondary**
    - Obsession with career progression will take care of itself in the **industry… tertiary**
  - Strong, ‘aware’ **leadership** is essential.
    - Keeping team together;
    - recognizing individual needs and excellent performers;
    - nurturing newbees,
    - considering diverse opinions,
    - facilitating contributions from everyone; make them feel important …
  - all are **essential** for an **effective** project manager.

4. Improving Automation Through Software Environments (1 of 3)

- The environment (tools) can dramatically impact productivity and effort – and thus schedule and cost.
- Huge number of available tools in marketplace for supporting a process.
  - Careful selection of the right combinations…
  - Recognize that these tools are the primary delivery vehicle for process automation.
- ☐ Mature software processes suggest that highly integrated tools and environment are necessary to facilitate the management of the process.

4. Improving Automation Through Software Environments – cont. (2 of 3)

i. Definition of the development and maintenance **environment** is a first-class artifact of a successful process.
   - Robust, integrated development!
   - Hire good people and equip them with modern tools.
   - □ A prime motivator: learning tools and environments of our profession!
- Yet today's environments still fall short of what they can be.
- So much more is coming…and coming… and…

4. Improving Automation Through Software Environments – cont. (3 of 3)

- Be careful of tool vendor claims.
- (Can prove anything with statistics!)
- □ Remember, the tools must be integrated into the development environment.
- □ Authors suggests that in his experience, the combined effect of all tools is less than 40% (5% for an individual tool) and most benefits are **NOT realized** without a corresponding change in the **process** that will require their use.
- But this is substantial!!

5. Achieving Required Quality (1 of 5)

- Many of the items we have discussed not only favorably affect the **development process** but impact overall quality.
  - (Remember this statement for questions ahead…)
- Author presents a rather comprehensive table – p. 49 – for our consideration:
- This table represents General Quality improvements realizable with a modern practice:

Table 3-5. General Quality Improvements with a Modern Process

Quality Driver	Conventional Process	Modern Iterative Processes
Requirements misunderstanding	Discovered late	□ Resolved early
Development risk	Unknown until late	□ Understood and resolved early
Commercial components	Mostly unavailable	Still a quality driver, but tradeoffs must be resolved early in the life cycle
Change management	Late in life cycle; chaotic and malignant	□ Early in life cycle; straight-forward and benign
Design errors	Discovered late	□ Resolved early
Automation	Mostly error-prone manual procedures	Mostly automated, error-free evolution of artifacts
Resource adequacy	Unpredictable	Predictable
Schedules	Over-constrained	Tunable to quality, performance, and technology
Target performance	Paper-based analysis or separate simulation	Executing prototypes, early performance feedback, quantitative understanding
Software process rigor	Document-based	□ managed, measured, and tool-supported

5. Achieving Required Quality – continued (3 of5)

- Additional overall quality factors:
  - ☐ 1. Focus on <u>requirements driving</u> the process – namely: **address critical use cases <u>early</u>** and **traceability late** in the life cycle.
    - <u>Balance</u> requirements, development and plan evolution
  - 2. <u>Use metrics / indicators</u> to measure <u>progress and quality</u> of the architecture as it evolves from a high-level prototype to a <u>fully compliant product</u>
    - Remember: the architecture drives much of the process! Discuss. What does it drive?
    - How do you think we measure this progress??

5. Achieving Required Quality – continued (4 of 5)

- Additional overall quality factors
  - 3. ☐ Provide integrated life-cycle environments that support early and continuous configuration control, change management, rigorous design methods, document automation, and regression test automation
  - 4. ☐ Use visual modeling and HLL that support architectural control, abstraction, design reuse…
  - 5. Continually look into performance issues. Require demonstration-based evaluations.
  - Think: HOW are these so? Can you answer??

5. Achieving Required Quality – continued (last)

- Performance issues:
  - Be **careful** with commercial components and custom-built components
  - Assessment of performance can **degrade** as we progress through our process.
  - **Planned, managed demonstration-based assessments – the way to go.**
    - WHY do you think this is so???
    - Particularly true to demonstrate EARLY architectural flaws or weaknesses in commercial components where there is time to make adjustments…

6. Peer Inspections: A Pragmatic View

- An old way ‘asserted’ to yield super results. -> Just don’t provide the return desired in today’s complex systems.
  - ☐ Good in some cases such as to nurture less-experienced team members
  - ☐ Catch the real bad blunders early.
- Inspections can be applied at various times during a cycle. Consider:

6. Peer Inspections: A Pragmatic View – continued (2 of 7)

- ☐ 1. INSPECTIONS FOR: **<u>Transitioning</u>** engineering info from one <u>artifact set</u> to another, thereby assessing the consistency, feasibility, understandability, and technology constraints inherent in the engineering artifacts.
  - Example: analysis classes will morph into design classes which will typically become part of packages or components… In doing so, **<u>have we lost</u>**, for example, the desired functionality?
  - If the functionality is accommodated by a number of design model elements, **can you ensure that the functionality is NOT lost**? Can you TRACE it?
  - Discuss.

6. Peer Inspections: A Pragmatic View – (3 of 7)

- ☐ 2. INSPECTIONS: Major milestone demonstrations
  - Force artifact assessment against tangible criteria for relevant use cases…
- 3. INSPECTIONS using Environment tools
- 4. Life-cycle testing – provides insight into requirements compliance…
- 5. INSPECTIONS: Study Change Management metrics – must manage Change and how change requests might impact both quality and progress goals.

6. Peer Inspections: A Pragmatic View – (4 of 7)

- Overall:
  - ☐ Ensure that critical components are really looked at by the primary stakeholders.
    - Cannot really look at all artifacts.
    - Inspecting 'too many artifacts' will not be cost-effective – on the contrary!
    - Many artifacts don't deserve / merit close scrutiny – and most inspections tend to be quite superficial.,

6. Peer Inspections: A Pragmatic View – (5 of 7)

- Love this: Many highly complex applications have demanding dimensions of complexity which include innumerable components, concurrent execution, distributed resources, and more.
- Most inspections thus end up looking at style and ‘first-order’ semantic issues rather than real issues of substance.
- Discuss technical / managerial reviews

6. Peer Inspections: A Pragmatic View – (6 of 7)

- Most major difficulties such as **<u>performance, concurrency, distribution</u>**, etc. are discovered through activities such as:
  - Analysis, <u>prototyping, and experimentation</u>
  - Constructing design models (can see requirements missing; can see architectural constraints unaccounted…)
  - Committing the current state of the <u>design</u> to an executable implementation
  - Demonstrating the current <u>implementation</u> strengths and weaknesses in the context of <u>critical subsets</u> of use cases and scenarios
  - Incorporating lessons learned back into the models, use cases, implementations, and plans.

6. Peer Inspections: A Pragmatic View - last

- Remember: the RUP is (among other things) architecture-centric, use-case driven, iterative development **process.**

- Iterations are **planned** to address (decreasing) priorities: risk and core functionalities as identified in Use Cases and Supplementary Specifications (=SRS)
- This iterative process evolves the architecture through the phases especially and mainly elaboration.

- Each phase has milestones and focus on inspecting critically-important issues.
- Overall there is very questionable ROI on meetings, inspections, or documents.
- Quality assurance is every stakeholder's responsibility