CS 430 - Lecture 07 - Teams I

Collin Roberts

September 27, 2023

Outline

Team Organization

To develop a software product of any significant size, a **team** is required.

Question

Suppose that a software product requires 12 person-months to build it. Does it follow that 4 programmers could complete the work in 3 months?

Answer: No:

1. There are new issues (communication / integration / etc.) once a team is involved, as contrasted with an individual.
2. Not all programming tasks can be fully shared in time or in sequencing. Maybe the software product naturally has three chunks, or maybe it has many chunks with complicated dependencies.
3. A project manager's **Gantt Chart** is a tool for managing the dependencies in a team project.

Team Organization

Definition 1

Brooks' Law *states that* **adding programmers to an already late project makes it later.**

Fred Brooks observed this phenomenon while managing the development of OS/360, for IBM 360 mainframes.
One reason (not the only one):
The more programmers there are on a team, the more communication paths there are, and hence the slower overall communication becomes.

Team Organization

Remarks:

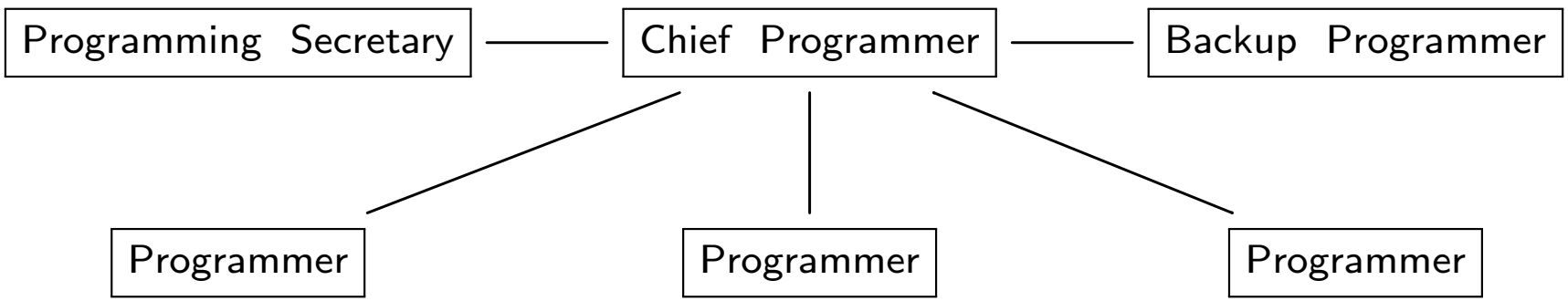
1. The rest of Chapter 4 focuses on team organization applied to the **implementation workflow**. The problems and solutions are equally applicable to other workflows.

Classical Chief Programmer Teams

A six-person team without a chief programmer has $\binom{6}{2} = 15$ communication paths. Every pair of people can communicate directly with each other.

Classical Chief Programmer Teams

A six-person team with a chief programmer (this is Figure 4.3 in the text) looks like:



There are only 5 communication paths.

Classical Chief Programmer Teams

Definition 2

A **Classical Chief Programmer Team** *is a team organized according to some variation of the above picture, possibly with fewer or more programmers, and having the following roles.*

Chief Programmer

- highly skilled programmer
- successful manager
- does architectural design
- writes critical/complex sections of the code
- handles all interface issues
- reviews the work of all team members (responsible for every line of code)

Backup Programmer

- needed in case chief programmer wins the lottery, gets sick, falls under a bus, changes jobs, etc.
- as competent as the chief programmer in all respects.
- does tasks independent of the design process (e.g. selecting test cases for black box testing)

Programming Secretary (aka Librarian)

- maintain the production library, including all project documentation
  - source code (responsible for compiling)
  - JCL (job control language, for running mainframe batch jobs)
  - test data (executes all tests)

Programmer

- They just program.

Classical Chief Programmer Teams

Strengths:

1. This has been enormously successful in a few cases. It was first used in 1971, by IBM, to automate the clippings data bank ("morgue") of the New York Times and other publications. If you have the text, see §4.3.1.

Classical Chief Programmer Teams

Weaknesses:

1. Chief/Backup Programmers are hard to find.
2. Secretaries are also hard to find.
    1. We seek someone with strong technical skills, then demand only clerical work from them.
3. The Programmers may be frustrated at being "second class citizens" under this model.

Classical Chief Programmer Teams

Remarks:

1. In reality, most team organizations lie somewhere between the two extremes of classical chief programmer (very hierarchical) and democratic (non-hierarchical).

Basic concept

Definition 3

egoless programming:

1. *Code belongs to the team as a whole, not to any individual.*
2. *Finding faults is encouraged.*
3. *Reviewers show appreciation at being asked for advice, rather than ridiculing programmers for making mistakes.*

Basic concept

Definition 4

A team of ≤ 10 *egoless programmers constitutes a* **democratic team**.

Possible **managerial issues**

1. For such collaboration to flourish, there must be a strong culture of open communication.
2. The path for career advancement may not be clear (by definition, a democratic team has no leader).

Strengths

1. Rapid detection of faults → high quality code.
2. Addresses the problem of programmers being overly attached to their own code.

Weaknesses

1. managerial issues as mentioned above
2. It is hard to create such a team. Such teams tend to spring up spontaneously from the "grass roots", often in the context of research as contrasted with the context of business.
3. A certain organizational culture is required before such a team can emerge.

Beyond Chief Programmer and Democratic Teams

These two team organizations sit at opposite ends of the continuum:

Classical Chief Programmer	**Democratic**
very hierarchical little individual freedom	little hierarchy much individual freedom

A Conflict Inherent in the Chief Programmer Model

1. The Chief Programmer must attend all code reviews. They are responsible for every line of code, as the Technical Manager of the Team.
2. The Chief Programmer must not attend any code reviews. They are the HR Manager, and reviews should never be used for HR performance appraisals (see Chapter 6).

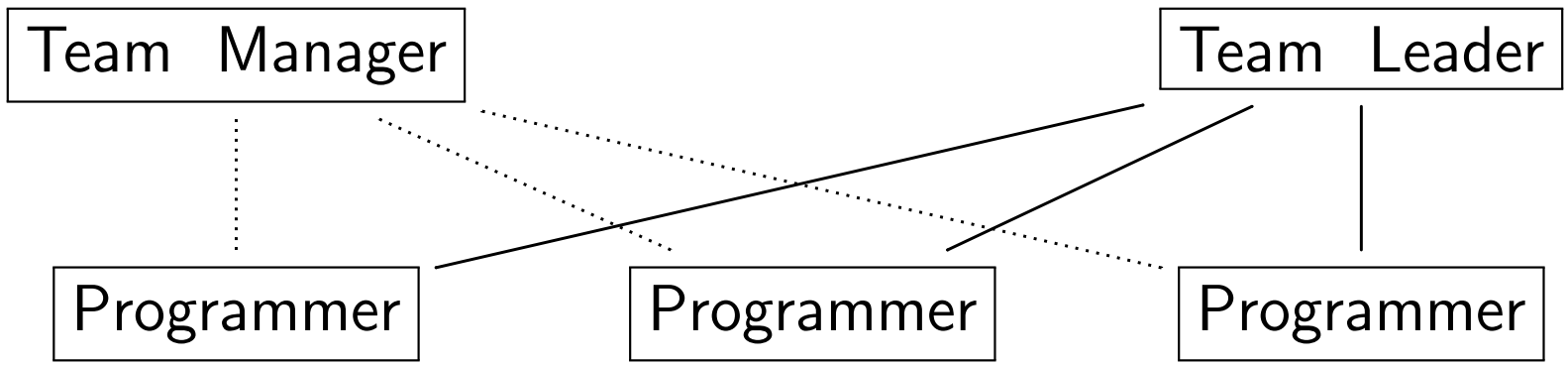
A Conflict Inherent in the Chief Programmer Model

Resolution

1. **Suggestion From Class:** Divorce the performance appraisals from review results (sounds good in theory; problematic in practice).
2. **From Text:** Split the Chief Programmer role into a Team Manager (non-technical) and Team Leader (technical).

A Conflict Inherent in the Chief Programmer Model

Requirement Clearly demarcate the duties of each role, wherever there could be some overlap. The picture below can be scaled up, as in Figure 4.5 of the text.



Student Questions

1. Do the Programming Secretary and Backup Programmer roles still exist here? **Instructor Answer:** In my experience, no:
    1. The first version of this model dates from 1971, when a program was a stack of punch cards. In today's environment, with "soft" code, and robust version control, the Programming Secretary is obsolete.
    2. While a backup programmer is no longer explicitly identified, every organization must grapple with succession planning, somehow.

Beyond Chief Programmer and Democratic Teams

2. Does the Backup Programmer role also have to be split up, like the Chief Programmer does?
   Instructor Answer: No, the split would occur when the Backup Programmer is promoted to Chief Programmer.

Beyond Chief Programmer and Democratic Teams

3. What are the strengths of the Classical Chief Programmer Team Organization?

Instructor Answer:

1. Key Observation: This team organization is extremely rigid with respect to which communication paths are permitted. Recall Brooks' Law (Definition 1). A Classical Chief Programmer is least vulnerable to the effects of Brooks' Law, because the number of communication paths only grows linearly as the number of programmers grows. Under a team organization in which any programmer can talk to any other programmer, the number of communication paths grows as the square of the number of programmers.
2. As will be suggested later for **Synchronize and Stabilize** teams, this team organization fosters a team culture in which all team members work together towards a common goal.