“Managing Multiple Projects”
A Book by
Michael and Irene Tobis

Delphine Khanna,
University of Pennsylvania
May 4, 2009

A Book

- "Managing Multiple Projects", by Michael and Irene Tobis, McGraw Hill, 2002.
- Just looked for a book on our topic (with good reviews on Amazon!)
- I am very happy with my finding
- Report on it
  - Focus on how it applies to the DL world

Usual bias toward single project

- Most management books & workshops focus on how to manage one big project.
- But in the real world many managers manage multiple projects at the same time
- And the “one big project” approach does not help at all with issues brought up by a multiple projects situation.

Very different approach

- Book very refreshing in that way
  - Very different approach from the corporate-oriented “Project Management 101” training, with mandatory use of MS Project, etc.
- Sometimes obvious common sense advice
- But overall a lot of food for thought and very practical advice

Target organizations

- Typical organizations that the book targets:
  - A printing shop, where one project would be to print an issue of a trade journal or a newsletter
    - Do all the graphic design and layout work, and then print out 5,000 copies
  - That one printing shop would be handling many such projects at the same time

Project size?

- Size of the projects described a bit smaller than the average DL project
  - Of course, I am not sure what "average DL project" means
  - Great variety of sizes
    - Ingesting one specific collection of 500 images into your DL repository
    - Vs. implement an entire Fedora-based architecture

Our position in the spectrum

- Most DL operations are probably somewhere between
  - the print shop --> one manager = many small projects
  - and a big corporation --> one manager = one single big project, or in fact, one part of a big project
- So we probably need to take inspiration from the two ends of the spectrum

Commitments

- Main issue with managing multiple projects:
  - The different projects compete for the team's attention
- Book's key concept: commitment
  - If you say that your team will complete project A by a specific date (or "in a timely manner"), this is now a commitment.
  - If your team has many competing commitments (i.e., many projects), what is the optimum way to manage them all?

Various types of commitments

- To complicate the matter further:
  - Big projects vs. small projects
  - Maintenance tasks, fixing a bug, etc. in a timely manner is also a commitment
  - Repetitive cyclical tasks
- So the competition is not only among projects of the same size, but among all kind of things:
  - big projects, small projects, one-off tasks, repetitive routines, etc.

Reliability

- Notion of reliability
  - Honoring *ALL* your commitments:
    - Big and small
    - High profile and low profile
    - New development and maintenance
    - Etc.
  - Goal: you don't want your team to be perceived as talented but unreliable
    - Releases a few great projects
    - But misses on other commitments
      - Especially smaller or lower-profile ones

How to be reliable?

- No overcommitment
- No commitments “falling through the cracks”
  - Especially the small ones
- No commitments treated as “second class citizens”
- Good general throughput

Steps to reliability

- Book: proposes a whole system and many specific recommendations to achieve reliability
  - How to assess your teams's workload and assess overcommitment
  - Concrete steps to get out of overcommitment. You will need to:
    - Increase throughput, or
    - Decrease demand
      - Including "how to say no"

Steps to reliability (2)

- Methods to estimate task and project length, how to keep track of your team's workload
- How to gather information from your team about ongoing projects and tasks
  - So that you get the information you need, but do not generate too much overhead.
- How to keep track of multiple projects and tasks
  - without heavy duty software like MS Project

Steps to reliability (3)

- How to build an “air tight” system where no commitment can ever fall through the cracks
- The different types of tasks (routines vs. one-offs vs. projects) and how to best handle each type
- How to maintain a good general throughput

Sorry!

- Sorry for not detailing: I have only 20 minutes.
- In the rest of the talk, focus on two particularly interesting concepts
  - Overcommitment (and how to avoid it)
  - Compartmentalization

Overcommitment

- The situation is bad when:
  - You have accepted too many projects
  - Each customer thinks that your team is working on their project actively and making good progress on it
  - But in reality, there is no way your team can work actually on all of them at the same time

Internal (hidden) prioritizing

- So what really happens is that you start prioritizing **internally**
- You choose to get to some of the projects first, leaving the others for "later"
- Customers whose projects are not worked on, see that their projects are making no or slow progress

  => Become very frustrated
- "Second-class citizen" commitments, whereas
  - **ALL COMMITMENTS SHOULD BE TREATED EQUALLY**

Internal (hidden) prioritizing

- This internal prioritization might not be a conscious and deliberate process
- It might just be that
  - you keep working on project A and B, and you keep **meaning** to get to project C as soon as possible
  - but this “as soon as possible” moment never comes or comes very late.

Variant

- Instead of doing internal prioritizing
- The staff tries to work on all projects at the same time, and everybody is spread thinner and thinner.
- As a result all the projects progress too slowly.

Time estimates

- One whole part of the book's solution is based on keeping time estimates on all your commitments in hours
  - All the way down to the specific task.
  - So that you know when your team is maxed out
  - And you can refuse new commitments
- Question: are there people who do systematic time estimates for every one of their team's tasks?
- Not for everybody?
  - Too much overhead?
  - Hard to do for R&D efforts?

A question of perception

- But I think that there can be other ways of avoiding the disconnect between
  - Projects committed to
  - And projects really worked on
- I would reformulate this more generally as question of perception
  - and therefore a question of communication

A question of perception (2)

- You need to find a system where your customers are never under the false impression that you are working on their project when you are not
  - Your team has to be working only on a reasonable amount of projects at a time
  - And it has to be crystal clear to everybody that this is the case

“Netflix List”

- “Netflix list” for our digital delivery projects at Penn
  - What is it?
  - It is clear to everybody that we are working only on the top of the list (a few **active** items)
  - And people know where they stand in the queue of our **future** commitments
  - Allows for **explicit external prioritizing**
  - Working very well for us
- Natural way to avoid overcommitment and hidden prioritizing, and to promote clear communication

Compartmentalization

- One of the very applicable solution that the book offers is the notion of compartmentalization
- Instead of having a single work queue for your team, create several of distinct queues

Compartmentalization (2)

- Particularly useful to take care of commitments that you have identified as "second citizens"
- Example of a company with a big account and other small accounts
  - 2 groups, one focused on the big account, and the other one on the small accounts as its first priority

Several ways

- Compartmentalization can be done in different manners:
  - Creating subgroups (e.g., big account vs. small accounts groups)
  - Or within an single individual's workload, e.g.:
    - 50% of time on project A and 50% on project B.
    - Morning on maintenance tasks and afternoons on new development

Explicit allotted time

- This is a great way to avoid 2nd class citizen commitments
- And therefore a great way to increase reliability
  - **You know that each type of commitments has explicit allotted time**

No miracles

- Of course it can't do miracles on its own
  - If you are committed to 20 projects and can only realistically work on 5, it won't help.
    - You need to solve the problem of over-commitment first.
  - If a specific queue has a slow throughput, progress on those commitments will be slow
    - You need to take care of any bottleneck

Compartmentalization at Penn

- Portfolios => clear areas of responsibility
- In the DLA project:
  - Separate teams based on media types
  - Each core programmers: 50% on DLA & 50% on Fedora
- New:
  - Core programmers for DLA: fundamental development vs smaller collection-driven development

Some other interesting points

- Acknowledgement that:
  - you need different levels of formalism for different levels of project complexity
    - MS Project is a useless overhead for small projects
  - and that growing organizations need to progressively implement higher level of formalism
    - But **NOT MORE THAN NEEDED**
    - Lightweight tracking systems are ok

Some other interesting points (2)

- Importance of tracking down bottlenecks (people and machines) and how to deal with them
  - Essential, because the whole system relies on the various work queues actually making progress.
  - (Not the same thing as bottlenecks in “single big project” situations)

Some other interesting points (3)

- How to recover from a crisis
  - If your team is seriously overcommitted, and suddenly everybody is in a state of crisis (stress, panic, etc.), what do you do?
  - No magic tricks
  - Excellent step-by-step guide on how to get out of the panic mode, and approach the situation rationally.

Some other interesting points (4)

- How to implement change in a group of human beings
  - Once you have decided on the changes you want to implement

Conclusion

- I am still digesting the information
  - To understand how it relates to my organization
- Happy to realize that some of the changes we have implemented over the years correspond to the concepts described
  - In such cases, conceptualizing things you have learned through experience is very useful.

Conclusion (2)

- Already found interesting ways of applying some of the concepts developed (e.g., some more compartmentalization)
- I highly recommend reading the book, if you want to dig deeper into the topic