Goal-Driven Project Management

Robert M. Keller
Director, Computer Science Clinic
Harvey Mudd College
2 September 2008

There are several traditional ways to manage projects, each involving various artifacts and tools. This document describes a method that the CS faculty has agreed to use for Computer Science Clinic 2008-2009. The reasons behind it are:

- We need a way to track progress of both projects and individual contributions. We need to know how each project is doing at any given time, and we need to know how to grade students based on individual contributions to the team effort. The proposed method, followed uniformly through the project lifetime and across all projects, would give us this information.

- We need a "lightweight" method that is both effective, yet does not extract too much in the way of administrative overhead. The proposed method seems to be minimal. Because it relates to problem-solving techniques in AI, it should be easy for CS students to understand and implement.

- While I am aware of no tool available that captures the method, development of such a tool could be a topic for a future software development project. Meanwhile, we will implement the method using the project tracker Trac and its wiki.

Goal-Direction

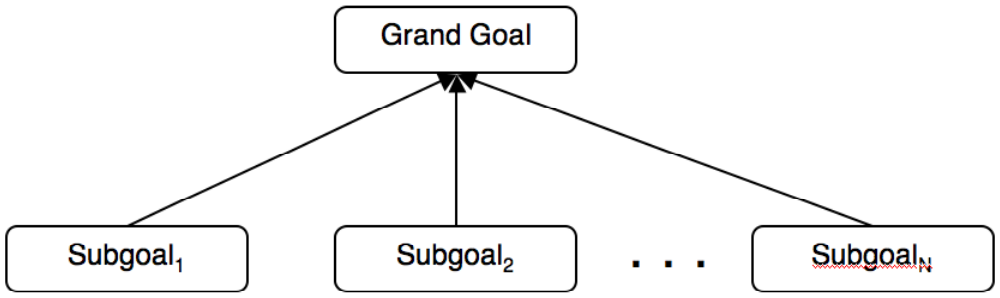
Every Clinic project has some kind of overall goal, although some are articulated better than others. However, even projects that are nebulous and exploratory should be able to develop meaningful subgoals during their lifetime. Projects should be evaluated based on how well goals and subgoals are achieved. Effort toward goals can be counted by breaking goals into smaller sub-goals that are either completed or not.

The **Grand Goal** of every project can be stated as achieving the project's objectives and deliverables. What is usually unknown at the start is how those goals can be achieved. Indeed, exploring ways to define and achieve subgoals is a form of **meta-goal**, and constitutes work, which should thus be counted. Likewise, the proper management of a project is a meta-goal for which credit should be given.

Goal Breakdown Structure

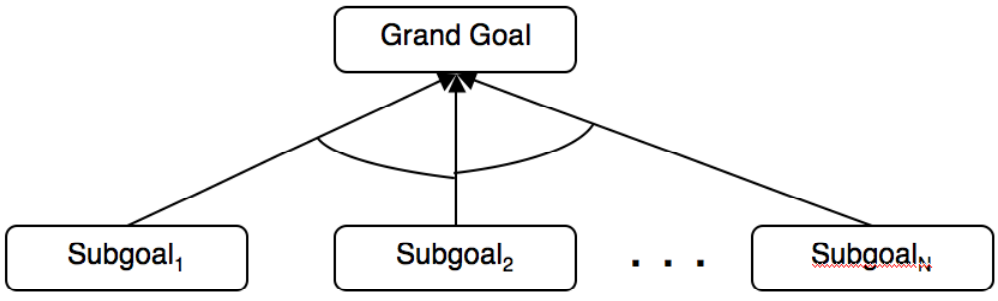
The Goal Breakdown Structure (GBS) is analogous to the traditional Work Breakdown Structure (WBS). However, rather than being fixed early in the project, it is created incrementally, by intent. Also, rather than being a tree, it is seen as a DAG (directed acyclic graph) for reasons to be explained.

The GBS is a DAG with the Grand Goal as its only sink (node with no arrows directed outward). Below the Grand Goal are various *subordinate* goals, having the Grand Goal as *parent*, as suggested below:



Subgoals that are subordinate to a parent goal

By default, subgoals of a goal are **AND-subgoals**, meaning that it is necessary to achieve *all* subgoals in order for the parent goal to be achieved. There may also be **OR-subgoals**, meaning that achievement of any one subgoal is sufficient to achieve the parent goal. OR-subgoals are shown be drawing a (geometric) arc across the arcs connecting them to their parent.

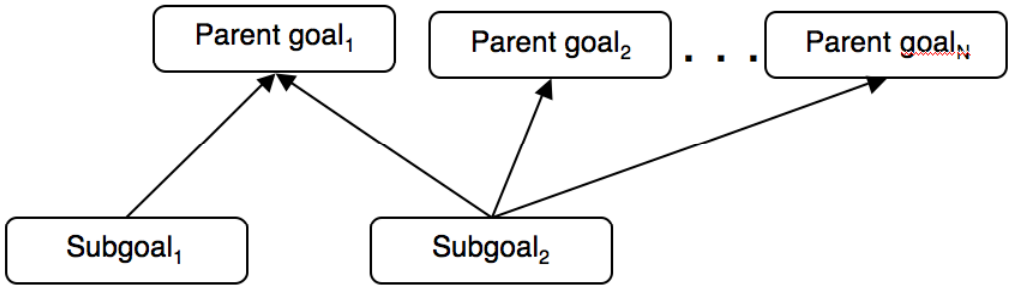


OR-subgoals that are subordinate to a parent goal

OR-subgoals are common in projects with an exploratory element. It is often useful to enumerate several options for achieving a parent goal, then assess their relative difficulty and risk factors. Deciding how to apportion work on OR-subgoals is an example of a meta-goal.

Shared Subgoals

Occasionally there will be subgoals that can serve more than one parent goal, either in the AND- or OR-sense. These are called **shared subgoals**, and will show up as nodes with more than one parent. When such subgoals are present, the GBS becomes a DAG and not a tree.



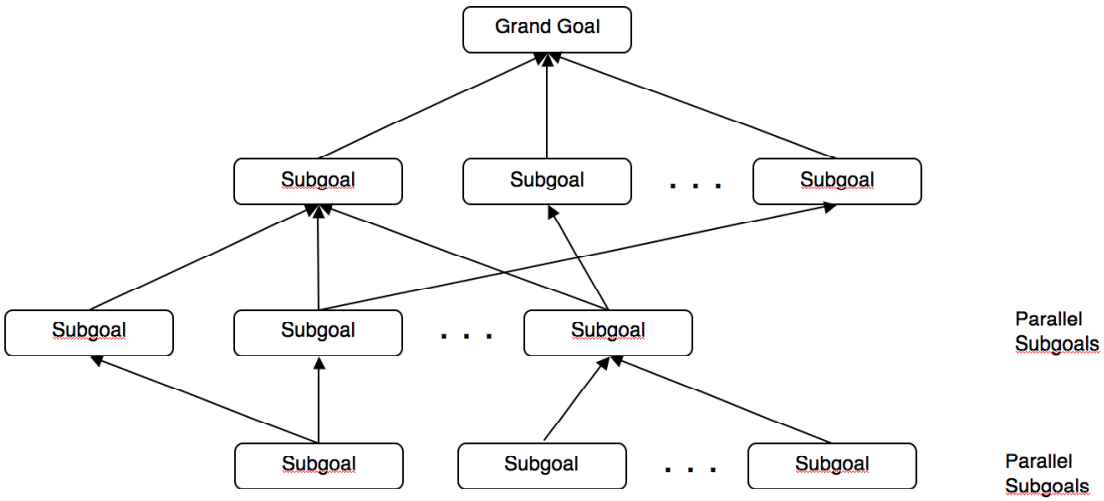
Depicting a shared subgoal

Critical Subgoals

A subgoal is called **critical** if the grand goal cannot be achieved without achieving the subgoal. For example, if there are no OR-subgoals, then every subgoal is critical.

Parallelism

Similar to a PERT chart, the GBS can show parallel subgoals, opportunities for team members to work independently or in smaller groups.



Parallelism

Goal Template

The team is responsible for self-managing its subgoals. This means that it will create the subgoals, schedule them, and monitor them for progress. Subgoals are created dynamically during the course of the project. Each subgoal will be documented using the following template:

Attribute	Meaning	Comments
Label	A shorthand way to refer to the goal.	Goals will need to refer to other goals, and labeling provides a convenience for doing this.
Description	A brief description of the purpose of this goal, in the context of its parents.	The description is written so that someone *not* on the team can also understand it.
Deliverables	Enumerates the work products that are going to exist when this goal is complete.	Sub-goal deliverables do not necessarily go to the sponsor.
Assignees	Persons who are working on this goal, or who should.	This is useful to determine work loading.
Status	This is trinary-valued: one of {**incomplete**, **complete**, **abandoned**}.	Rather give credit for partial completion, break the goal down into sub-goals that can be assessed individually.
Priority	A number indicating the estimated importance of achieving this goal (1: high to 5: low).	The priority is used in making staff assignments in the short term.
Target date	The date by which the goal is supposed to have been achieved.	The target date is used to assess whether the project is having difficulty or is on schedule.
Start date	The date on which work on the goal actually began.	The start date helps determine whether there are goals that are attention starved.
Time Allocation	Estimated total person hours to accomplish this goal.	If there are shared subgoals, the time might be shared among them.
Time Spent	Actual person hours used to accomplish this goal (approximate).	Recording historical data can help improve prediction.
Creation date	The date on which this entry was made.	For record-keeping.
Creator	Person who created this goal.	For questions.
Parents	The labels of the parent goals.	These are useful in showing the DAG and determining criticality.
Subordinates	The labels of the subordinate goals.	

The filled in templates will be kept on the project tracker/wiki. It is expected that the templates can be “rubber-stamped”, even partially filled in, to minimize the overhead in creating them.

Tool Automation

No tool is available currently, but some attributes of such a tool might include:

- Version control: Automatically track revisions to the GBS.
- Criticality assessment: Highlight goals that are more important.
- Time assessment: Compute the critical path and staff-loading hours.
- Completion assessment: How near to completion is the project.

Typical Clinic Goals (partial list)

- Research relevant topics
- Interact with sponsor
- Create Statement of Work
- Identify supplier sources
- Investigate applicable code libraries
- Prepare phase n presentation (n = 1, 2, 3)
- Create midyear report
- Design software
- Test software
- Document design
- Document software
- Document testing
- Create final report
- Create final poster
- Complete CDROM directory
- Deliver software to sponsor
- Meta-goals, such as maintaining the GBS itself

Summary

No management method is entirely free of overhead. The proposed method eliminates some overhead inherent in traditional methods. Because it is incremental, it fits well with “agile” methods that are increasingly popular. Because goal structures are based on computer science concepts, it is hoped that this method will have more appeal than some of the more traditional ones.