**Project Plan Guidelines**

A project plan is the most important, yet often neglected artifact for the typical software development project. A project plan is a sufficiently complete, ***professional*** document for communicating information to software engineers and management needed to understand what the project entails, how it will be produced and controlled and what the effort and schedule estimates are for the project. The project risks, quality focus, and support needs are specified as well.

**Key Components of a Project Plan:**

* **Overview** three to five paragraphs describing product function, platform, customers, schedule and development responsibility.
* **Goals and scope** - what's in scope, what's out of scope?
* **Deliverables** high level releases and content.
* **Risk Management** identification of risks, mitigation strategies, how they will be managed.
* **Scheduling and estimates** work breakdown structure, overall project schedule, resource allocation, estimation techniques used (justify estimates), how will project be tracked, how will schedule changes be made.
* **Measurements & Metrics** what measurements will be collected, what metrics will be created, how will they be used, why were they chosen. These measurements will form the basis for the project's quality assurance plan and drive process improvement.
* **Technical Process** what methodology will be used, what tools and techniques are required, what internal artifacts are required to be maintained. (This section typically references other development documents requirements, design, test plan, etc. The SDP identifies what artifacts are to be created and how they will be maintained.)

**References:**

* Rapid Development by Steve McConnell (available on [Books 24x7](https://ezproxy.rit.edu/login?url=http://library.books24x7.com/library.asp?%5EB)) see Chapter Seven for background on software development life-cycle models.
* Construx Software [templates & examples](http://www.construx.com/cxone/basic/map.php) (Steve McConnell's company) see "Engineering Management Section". Note that you will need to create a free login for this site. There are many very useful document templates and checklists available here.
* [Planning and tracking spreadsheet](http://www.se.rit.edu/~swen-561/ProjectResources/ActivityTracker.xlsx)
* [Risk management spreaSearchdsheet](http://www.se.rit.edu/~swen-561/ProjectResources/RiskManagement.xlsx)
* As part of the Microsoft Academic Alliance, you may download a free copy of Microsoft Project.

# Overview

The software allows users to create client records (single, couple, etc) with personal information such as date of birth. There are essentially two modes (pre-retirement where clients are accumulating assets and retirement where clients are spending down those assets).

Pre-retirement inputs would include savings rates in various types of accounts (IRA/401k, Roth, after-tax), and expected rates of return. There are other parameters depending on the scope of project – see next section. Clients and other users would estimate projected retirement dates and the project software would calculate expected savings amounts in each account type.

During retirement clients derive their income from multiple sources with varying tax implications. Social security, IRA distributions, Roth distributions, pensions, and other savings all have different tax treatments. The goal is to optimize the annual withdrawals to maximize the number of years assets last. If there are substantial assets, another goal could be to maximize the assets remaining after a 30-year time horizon when the client could be expected to pass away. The optimization can extend to pre-retirement years. For example, if the client is 50 years old they need to know if Roth savings, 401k savings, or other savings would be most beneficial for the long-term.

This tool will be used internally by Beltz-Ianni & Associates. It will be run on a local server for use through a web interface. Utilizing this application will allow Beltz-Ianni employees to optimize the savings and spending patterns of their clients with respect to tax rules and regulations. These employees have experience with similar money management software, but will have varying degrees of technical experience.

 This project will be ongoing for the Fall 2016 semester and Spring 2017 semester. The team will consist of two developers who work on the front-end and two that focus on the back-end.

# Goals and Scope

TERI can be broken into the following modules:

## **Income Tax Module (Tax Engine)**

This module essentially calculates the client’s 1040 for each year. Inputs would include earned income, investment income, social security, other income, 401k contributions, personal exemptions, mortgage interest deduction (based on entered mortgage information), property tax deduction, charitable deductions, and other deductions. The mortgage interest deduction is calculated by the software. Most of the other deductions are entered and adjusted for expected inflation each year. State income taxes should be estimated as a project simplification.

The tax engine will be robust enough to calculate correct outputs for the test data provided by the sponsor. It will not calculate all portions of the 1040.

**Spending Module**

Calculates how much to draw from each asset based on long-term sustainability of assets. An added complexity is determining when to start social security income. Given a client expects a relatively long life delaying social security is often recommended. Clients may want to start social security at a specific age for a number of reasons so the projects needs flexibility in this area.

**Optimization Module**

(Basic) Calculates savings expected at retirement. For non-retirement accounts, the system should create a tax liability each years based on assumed dividends and capital gains.

(Advanced) Given a client’s current age and financial situation this module determines the optimal amount to save in each account type. Current tax situation and results of the spending module would be key inputs.

The basic variation will be implemented for this project. The advanced variation is out of scope unless there is additional time.

**Report Module**

(Basic) A table showing clients results of various savings rates and the improvement in retirement outcome by changing from current plan to “optimized plan”

(Advanced) Graphs and bar charts showing clients results of various savings rates and the improvement in retirement outcome by changing from current plan to “optimized plan”

The basic variation will be implemented for this project. The advanced variation is out of scope unless there is additional time.

##

# Deliverables

**Application** - A software system that can be installed on the Sponsor’s computers.

**Documentation** - Beside general code documentation, optimization strategies, tax rules and how they were implemented, along with any other strategic things need to be documented.

**Website** - Senior Project website.

# Risk Management

Attach Risk Management Spreadsheet.

# Scheduling and estimates

Estimation will done using “commitment hours”. The student team will have a pool of committed hours which can be allocated for user story estimates for each sprint.

The Scrum process methodology will be used. See technical process for more information.

# Measurements & Metrics

**Velocity** - Used to measure how much work is being completed. Story points will be assigned to each user story (feature). The number of story points completed each sprint is the velocity. The velocity can then be used to accurately estimate what can be completed in the next sprint.

**Testing Code Coverage** - Amount of code covered.

**Cyclomatic Complexity** - Measures the structural complexity of the code. It is created by calculating the number of different code paths in the flow of the program. A program that has complex control flow will require more tests to achieve good code coverage and will be less maintainable.

Jira will be used for work tracking

# Technical Process

The Scrum process methodology will be implemented for this project. We will also utilize pair programming.

## **Sprint 0**

## Sprint 0 will be the initial sprint where the student team will construct the product backlog based on the sponsor’s requirements.

## **Sprints**

Sprints will be 2 weeks long. They will begin on a Thursday after the spring planning meeting (5PM-6:15PM), then end on the Tuesday 12 days later at the sponsor meeting (5PM-6:15PM)

## **Sprint Review Meetings**

On the Tuesday sponsor meeting at the end of every sprint, the student team will present the product as a result of the current sprint. The sponsor and coach can give feedback during this meeting which will be taken into consideration for the next sprint

## **Sprint Retrospectives**

Discuss feedback from the sprint review meeting and consider making necessary adjustments for the next sprint

## **Sprint Planning**

Student team populates the sprint backlog for the upcoming sprint.

## **“Daily” standup meetings**

The students will discuss what they did, what they plan on doing, and any roadblocks at every Thursday meeting.