Connor Becker
CS460 – MyIPFWAdvisor
15 February 2012

Two entities

- Requirements
  - Root
    - Top level requirement for each degree path (BS, BA, etc.)
  - Branch
    - Areas of requirement (Core CS, Supporting, Concentrations, Informatics)
  - Leaf
    - Individual Requirements to be satisfied by particular satisfiers (classes, transfer credits, etc.)
  - Each requirement would contain the fields:
    - Description
    - Signature (a unique identifier)
    - Known Satisfiers (empty except for leaves)
    - Business Rule (rule pertaining to completion of sub-requirements)
    - Sub-requirements (empty if leaf, otherwise contains sig's for each constituent requirement)
- Satisfiers
  - Each satisfier would contain the fields:
    - Description
    - Course ID (filled if the satisfier is a class or maps to a class)
    - Signature (unique id)

The system would hold the requirements template as a tree of objects that could be rebuilt from a database table whenever the system is restarted or is instructed to do so. The satisfiers would likewise be stored in a database; when a student logs in, the system grabs the class information and builds a collection of satisfiers for the student based on the student's academic achievements.

The system can then traverse the requirements tree, taking note of requirements that have not been satisfied. From that list of unsatisfied requirements, a list of satisfying classes can be built that can be handed off to sugar.

"Building a bingo sheet" in this system would therefore require building not only of requirements, but also building of satisfiers: the requirement "Learn the fundamentals of programming using Java" would be built, but it would require that one builds the satisfier that might be called "CS 160 – Introduction to Computer Science I", with a course ID indicative of a course called "CS160". It would be helpful if the system would check to see if satisfiers similar to the builder's request already exist in order to expedite the process.

This would be a generalized system of reconciling prerequisites. Since not all satisfying requirements can be known beforehand (transfer credits from institution x, do they count as course q?), satisfiers could be built for these previously unthought-of situations, and requirements could be updated accordingly—all without having to redesign any programming code.