Selection of software components

Large software systems consist of a complex architecture of interdependent, modular software components. These may either be built or bought off-the-shelf. The decision of whether to build or buy software components influences the cost, delivery time and reliability of the whole system, and should therefore be taken in an optimal way.

Consider a software architecture with n component slots. Let I_i be the set of off-the-shelf components and J_i the set of purpose-built components that can be plugged in the i-th component slot, and assume $I_i \cap J_i = \emptyset$. Let T be the maximum assembly time and R be the minimum reliability level. We want to select a sequence of n off-the-shelf or purpose-built components compatible with the software architecture requirements that minimize the total cost whilst satisfying delivery time and reliability constraints.

The parameters of this problem are:

1. for all $i \leq n$, s_i is the expected number of invocations of the component within the large software system;
2. for all $i \leq n, j \in I_i$, c_{ij} is the cost, d_{ij} is the delivery time, and μ_{ij} the probability of failure on demand of the j-th off-the-shelf component for slot i;
3. for all $i \leq n, j \in J_i$, $\bar{c}_{ij}$ is the cost, t_{ij} is the estimated development time, τ_{ij} the average time required to perform a test case, p_{ij} is the probability that the instance is faulty, and b_{ij} the testability of the j-th purpose-built component for slot i.

The probability that no failure occurs during the execution of the i-th component is

$$\varphi_i = e^{s_i \left(\sum\limits_{\substack{j \in I_i \\ j \text{ assigned to } i}} \mu_{ij} + \sum\limits_{\substack{j \in J_i \\ j \text{ assigned to } i}} \vartheta_{ij} \right)},$$

where

$$\vartheta_{ij} = \frac{p_{ij} b_{ij} (1 - b_{ij})^{(1-b_{ij})N_{ij}}}{(1 - p_{ij}) + p_{ij}(1 - b_{ij})^{(1-b_{ij})N_{ij}}},$$

and N_{ij} is the number of tests to be performed on purpose-built component $j \in J_i$ and slot i.

Formulate this problem as a mathematical program, reformulate it as a Mixed-Integer Linear Program (specifying what basic reformulations you have used, and whether they are opt-reformulations, narrowings or approximations) create a simple instance and solve it with CPLEX.