METAII - MIT Rhodes+Ross
Software Code Description– 9/28/11 - Donna Rhodes. Adam Ross, and Matt Fitzgerald
This document presents a breakdown of the VASC process into the relevant functions of the VASC code base, with short descriptions on how to use the complicated functions.

1. Set up data for Epoch-Era Analysis
****

The functions that set up the MATLAB workspace for EEA are located in the Data and Data Loaders folder. These functions are highly specialized for whatever form the data input takes, and thus are not generalizable between case studies. When adapting VASC to your own case, pay special attention to creating these files to process your own data!

Space Tug
The key file here is dataLoader.m. The other files in the Data and Data Loaders folder are the underlying input data (mat, csv, and xls files) and other functions that process them and are called in dataLoader, which assembles much of the necessary information in the MATLAB workspace.

In order, dataLoader does:
1) calculates attributes and costs for each design in each context (via designCalculator)
2) calculates multi-attribute utility for each design in each epoch (spaceTugMAUcalc)
3) creates (design x epoch) matrices for cost and MAU
4) calculates (design x epoch) Fuzzy Pareto Numbers (via calcFuzzyParetoNumbers)
5) loads the transition matrices (with commented out code for creating them originally)

dataLoader is a script and therefore has no inputs, but calls on two data structures to do that work: tradespaceData.csv (to create the full factorial design vector enumeration) and spaceTugPrefSets.mat (to get the utility curves needed to evaluate MAU). These are the sorts of inputs and outputs you should expect to code for if applying VASC to a multi-attribute tradespace study. Other structures may prove relevant for different types of case studies, but the key outputs here are the (design x epoch) matrices that form the inputs to most of the rest of VASC.

X-TOS
This case also has a dataLoader.m file, but is obviously different. In this case, the data for X-TOS comes not in the form of design variables and preference curves, but rather pre-calculated single- and multi-attribute utilities, and dataLoader is a function that calls these presaved data files and concatenates them into the (design x epoch) matrices and specifies a few other useful constants (number of designs/rules/epochs)

2. Define Changeability Rule Usage Strategies
****

This step is handled in the Strategies folder. Each function in these folders determines the selected transition path (if any) for each design in each epoch, reading in the transition matrices and any other relevant information necessary to make that judgement. For example, the maxU (maximize utility) strategy functions also take MAU as an input, along with acceptable cost/time thresholds for the transition. The outputs of these functions are three (design x epoch) matrices: one for the target design after transitioning (endStates), a cell array for the change mechanism path taken (rulesExecuted), and another for the total [$ time] cost of the transition (transCost).

The general information flow inside these functions is:
1) determine potential end states from transition matrices
2) rank and sort end states via decision criteria
3) test all potential paths for the best end state, choose least expensive
4) if too expensive, proceed to next best design
5) if all transitions are too expensive, signifies no transition (endState==0)

Space Tug
There are 3 strategies for space tug: maximize utility, maximize efficiency, and survive (change only if design is infeasible in current epoch). A fourth strategy (maximize profit) is included in the era analysis but NOT in this phase, as profits depend on epoch lengths which are not defined until an era is created.

X-TOS
Because of the nature of X-TOS data set, which includes only designs always feasible, the survive strategy is omitted here.

3. Multi-epoch Changeability Analysis
****

Having populated the workspace with the basic EEA constructs and the strategy results, we now have all of the data necessary for performing multi-epoch analysis. A variety of functions were created to assist this process, and they are located in the Helper Functions folder. A sample:

- calcFPS(endStates,fuzzyParetoNumber) calculates the fuzzy pareto shift for each design in each epoch
- FPSdistrPlot(FPS,designList) then plots the above data in distribution form for a set of designs of interest specified by the second input
- calcEffFPN(FPN,endStates) takes the FPN matrix and replaces the values for design-epoch pairs in which a transition occurred with the updated FPN. This can be summed horizontally (across epochs) to calculate effective (fuzzy) pareto traces
- ruleExecutionCounter(rulesExecuted,numRules) tallies the total number of transitions using each change mechanism, which can be used to calculate a design's likelihood of using a rule at any given epoch switch (via ruleLikelihood.m)

4. Era-level Changeability Analysis
****

The Era folder contains functions that construct a sample era and then, given an initial design, model the strategic changes performed by the system. They also record any era-level performance metrics deemed of interest. Each function call runs ONE era, so in practical use these functions are called in loops to sample thousands of eras.

<u>Space Tug</u>
The Era Simulation folder contains 4 functions, one for each strategy. These are slightly unusual in that they *do not take the end states determined by the strategy as inputs*. That is because Space Tug has a quirk where the stakeholder knows the length of each epoch as it begins (since the epochs are modeled as contracts), and this knowledge affects the choice of transition. Thus, each of the 4 functions is the same except for the section which determines the transition of choice similarly to the Strategy functions, except making sure that the transition time does not exceed epoch length. Note the large number of outputted metrics: Revenue, Cost, Transition Cost, Rules Used, Utility Months, Best FPN, Worst FPN, Average FPN, Average FPN not counting infeasible epochs, Met/Missed contracts. All of these can then be compared between designs of interest while performing era analysis.

<u>X-TOS</u>
The era functions for X-TOS are simpler: they simply take an initial design and the predetermined strategic transitions as inputs and then walk through the constructed era. This is more typical of what you would expect a VASC era simulator to look like. There are 3 functions in the Era Modeling folder which differ in the way they determine epoch length: geometric distribution, poisson distribution, and constant duration. These are included simply to show how you can add the variable epoch length wrinkle quickly and effectively to VASC.

Demo Scripts
****

Each case study comes with a set of demo scripts. These are labeled by strategy and perform a basic start-to-finish VASC process for the given strategy. View these as outlines for applying all of the above functions or for constructing your own case studies.

Other Helper Functions
****

Not all of the helper functions were mentioned above; the remainder are listed here with their basic function for your use:

- bestParetoTraceFinder – given FPN and a fuzzy threshold, this function finds all of the designs with the BEST fNPT
- caleFOD – calculated filtered outdegree for each design
- cellSpy – makes a spy() plot, but works on cell arrays: useful for visualizing the collapsed transition matrices
- gen_pareto_set – finds the pareto set when given a list of designs with their objective functions. Can do fuzzy pareto sets as well.
- plotFNPT / plot FOD – makes the fNPT/FOD by design number plots useful for identifying designs of interest
- ruleCollapse2 – this collapses transition rules saved in Cost{rule} cell arrays into a single cell array. It also evaluates multi-arc transitions. designPathTool is a subfunction used to fully enumerate all possible transition paths.
- transitionAnalysis – this function is a work in progress, but it analyzes the end states determined by a strategy and outputs sets of designs with different characteristics. It will tell you what designs are ever targeted by a transition, what designs never become infeasible when considering changeability, and what designs never transition. It will also find "implementation families", which we define as steady-state loops of designs when applying a particular strategy. That is, if three designs do nothing other than stay the same or change to one of the other two, those three are in the same "implementation family" because once you transition into the family, you won't leave. Future SEAri research will probably be spent looking into the properties of these.