Software Description for Valuation Approach for Strategic Changeability

META-II Technical Area Two:
Metric of Adaptability for Cyber-physical Systems

Contract FA8650-10-C-7084

Donna H. Rhodes
Adam M. Ross
Matthew Fitzgerald

Massachusetts Institute of Technology

September 2011

This software description serves to provide an overview of the provided software.

VASC Software Description

- Summary / Documentation by Matt Fitzgerald

1. Setup / Necessary Constructs

Remember that the first step of VASC is setting up your case study into an Epoch Era Analysis structure, with a defined design space, epoch space, and transition rules. For more information on this, refer to the SEAri final report.

The most common constructs are the results of objective functions which are calculated for each design in each epoch (e.g., MAU, Cost, FPN), which should be calculated and stored in (design x epoch) matrices, and are frequently used to look up these values.

Transition rules are stored as (design x design) matrices or cell arrays. Typically, before they are collapsed, each rule is stored as its own (design x design) matrix in a (1 x rules) cell array, with a zero corresponding to no change and a 1 corresponding to an available change from the row design to the column design. Transition costs (dollars, time, anything) can be stored either in multiple copies of these matrices, as they are in our examples, or in some other lookup table. The Rule Collapsing process (see: ruleCollapse.m) reduces these down to three (design x design) arrays, where each cell contains another matrix or cell array listing all of the Pareto Efficient paths from the row design to the column design (in the big array). There is one array each for Rule Paths (ordered execution of rules from here to there), Design Paths (the intermediate designs traveled on that rule path), and Transition costs (the total cost along all cost dimensions for traveling that path), and all of the efficient paths between a design pair have their own row of the subarray in that cell.

2. Information Flow (I/O)

The general path of the data in this code base is:

A) Transition Rules / Objectives --> Strategy simulator --> Strategic

transition execution matrices
This step takes the defined transition rules and determines which rule path will be executed from a given design in a given epoch. The Strategy simulator is a function which reads in the rules and the data for any objective function relevant to the path decision, and determines which target design and transition path is the best choice for each design/epoch pair according to some pre-constructed logic (e.g., maximize MAU for less than X cost). A DIFFERENT STRATEGY SIMULATOR FUNCTION IS NEEDED FOR EACH STRATEGY UNDER CONSIDERATION, unless you prefer to make a single function with an input that determines which strategy to use that activates a different loop depending on the selection, but I find this to be cluttered. There are 3 outputs to the strategy simulator, all of which are (design x epoch): a matrix of the chosen design # of the transition end state, and a cell array for the chosen rule path and transition cost. The design path that is stored as an output of the rule collapsing process can be saved as well, but is not typically used outside of the strategy simulator (which may choose to set a constraint against using certain designs as intermediate steps).

B) Strategic transitions --> Multi-Epoch Analysis Helper Functions --> Valuable Changeability Metrics (FPS, ARI, etc)
With the transitions determined by the strategy, other functions process that data into metrics used to value the design's changeability. These metrics will also come in (design x epoch) form. Other functions are used to plot or display this data effectively for analysis in the VASC process.

C) Strategic transitions --> Era simulation function --> Sample eras and era statistics
Again, knowing what transitions will be selected under a given strategy allows further analysis, this time via the generation of sample eras. An era simulation function typically starts at an initial design, uses a helper function to generate successive epochs, and then reads the strategy outputs to determine where to transition, while recording useful era statistics (transitions used, rule usage frequency, FPN or utility performance over time, etc).

3. Helper Functions

Helper functions refer to the large collection of miscellaneous function that serve to assist in the analysis process outside of determining the results of a strategy or simulating an era. Remember: VASC is fully customizable to whatever metrics or objective functions you wish to use, as long as it is meaningful to calculate them for design/epoch pairs. Thus, not all of our helper functions may be of use to you. However, here are a few basic types and corresponding examples from our code base:

"Calculate" functions (ex, calcFPS, calcARI, others): It is good to have functions dedicated to calculating metrics on (design x epochs) matrices, as you will be calling them frequently.

Plotting function (ex, plotFPSdistr): VASC is still designed to be used by an operator to explore value, and thus must be able to display the unwieldy amounts of information it generates in such a way that comparisons between designs or transition rules is understandable at a glance to a user. Automated plotting functions allow a user to bring up a visual of a lot of data quickly, allowing for faster and better understanding of the problem.

The ruleCollapse.m function is also a helper function. This function takes uncollapsed transition matrices (1 x rules cell array) and enumerates all of the paths from each design to each other design, saving only ones that are Pareto efficient along all of the transition cost dimensions.2

4. Create Your Own

I've already mentioned that the VASC process itself is customizable, and thus you should feel free to swap in metrics or objectives relevant for your application. The generic (design x design) transition matrices and (design x epoch) objective matrices are the standard forms that we use at SEAri, as we have found that they are intuitive to work with, but it is feasible to make an extensive hash or lookup table.

If you are trying to convert your own data into a VASC analysis using this code, the most critical step is in creating the "Necessary Constructs" of section 1. You should create a script that takes you existing data and puts it in the forms expected by the rule collapse, strategy, era, and helper functions. For an example of this, see the dataLoader.m file in the Space Tug case study example. This is the first script called at the beginning of any analysis, and puts all of the relevant cost, utility, and transition information into the MATLAB workspace.

5. Unusual Situations

Not all case studies fit into a nice mold, and VASC is no exception. You may run into cases that prevent the use of our basic code framework. For example, we have already encountered something like this with the Space Tug case study, in which the epoch duration is KNOWN to the decision maker (an atypical situation derived from treating epochs like contracts), and thus can be used to inform any strategic transition decisions. Under this assumption, strategic transition decisions can not be made independently of era simulation. We ran some

strategies in the Multi-epoch context by assuming that the epoch duration was long enough to accommodate (and justify) the time delays of any change. For the era simulation though, we needed a different era sim function FOR EACH STRATEGY, as the strategy loop to determine the best path had to be run live. Refer to the Era Simulation folder of that case study to see how we implemented this. It is possible that you will have to / want to make similar adjustments to the code flow for your case study.

6. Example Cases / Demos

This code is implemented on 3 case studies to serve as examples of VASC on problems of varying complexity: Space Tug, X-TOS, and SRS.

A) Space Tug

Space Tug is a small EEA study (384 designs, 16 epochs) investigating potential designs for a satellite designed to pull other satellites in orbit. The epochs are treated as "contracts" for different users/purposes, motivating dramatic changes in preference ordering of the attributes. The era construction is relatively simple, with only a single forward-moving context variable (technology level) in addition to the 8 preference sets. As mentioned in the previous section, a side effect of the "contract" story is that the epoch end times are KNOWN to the stakeholder, which may affect his decisions about design transitions.

Code Organization (by folder)

Data and Data Loaders: the function dataLoader.m loads the .mat files included in this folder to create the tradespace and collapsed transition matrices (the uncollapsed transition matrices and the collapsing code is included for your viewing, but the collapsed matrices were saved after calculation to save time).

Demo Scripts: this folder includes scripts that demonstrate the proper order to execute the functions in for the generation of VASC data and analysis. There is a script for each strategy. This is a good place to look in order to see fast results, or to become accustomed with the data flow and use of the functions.

Era Simulation: each function in this folder runs a sample era for a given initial design. As mentioned above, the nature of the "contracts" results in the need for a different function for each strategy. As used in the demo scripts, these functions are typically looped thousands of times to get average era performances/statistics for each design.

Helper Functions: here you will find the multitude of supporting functions for the other scripts. In addition to the previously mentioned "calculate", plotting, and ruleCollapse functions, you will also find

ruleExecutionCounter: reads strategy output and reports frequency of transition rule usage

gen_pareto_set: reads a list of objective values and reports the (fuzzy) pareto efficient designs

designPathTool: this is used by ruleCollapse in order to enumerate all possible paths

cellSpy: a utility for seeing spy() plots on the elements of a cell array, good for finding patterns/densities in transition matrices

bestParetoTraceFinder: reads in FPNs, reports design with the best (fuzzy) NPT

Strategies: these functions perform the standard multi-epoch decisions of which transitions to use for each design in each epoch. There is a different function for Maximize Utility, Maximize Efficiency, and Survive, as the logic is different for each one. For this case study, despite knowing the end times of the epochs while in an era, we simply assume that the epoch is long enough to accommodate any desired transition. Note that there is no function for Maximize Profit in here, as profit is tied in completely with epoch duration and thus era analysis.

Utility: this folder contains an MAU and SAU mat file for the tradespace, along with a function (called by dataLoader) to calculate these from the design variables.

B) X-TOS

X-TOS is a medium-to-small EEA study (~3000 designs, 56 epochs) investigating potential designs for a low-atmosphere sampling satellite. This is a very basic EEA setup, where the epochs are merely perturbations on the elicited DM preferences, to find what designs can effectively deal with any sensitivity to preference. It is the most basic implementation of our code.

OLD CASE - AWAITING FULL RETCON TO VASC FOR CONFORMANCE

C) SRS

SRS is a large EEA study (~23000 designs, 972 epochs) investigating potential satellites/constellation for a satellite-based radar system. It is a complicated era structure, with 8 different context variables, 3 of which are forward-moving only, and tracking of the design through the Design, Build, Test, and Operate phases of development, which affect available transition rules and costs.

NEWEST CASE – IN PROGRESS