Focus Software: Process and Results

Jennifer Malm, Suzanne Childress, Erik Sabina, DeVon Farago, Jerry Vaio, Scott Meeks
Software Goals

• Explainable and understandable
• Replicable and maintainable
• Scalable
• Transferrable
• Distributable
• Upgradable
• Integratable
• Tunable
A few design comments

- Breadth-first design
  - Runs one model component at a time
- Loosely-coupled services and components
  - Easier distributability
- Highly object-oriented code
  - Enhances several design objectives
- Database driven
  - One of our favorite parts!
Utility function structure: database
Utility function structure: C# classes
Output data: database
Running it

• Tests on 1, 2, 4 and 8 processors
  • Doubling cores cuts run time in half
• Run time – 48 hrs
  • One db server, one code, 4 cores each
• Design effects on run time:
  • 10 highway time periods, 4 transit.
  • 2,800 zones, no location choice set selection (yet.)
  • “big” OO language (C#)
• Exploring hardware options
  • Outsourcing
  • In-house computing
Explainable and Understandable

```csharp
log.InfoFormat("Starting person loop @ (0) with (1) rows", DateTime.Now, inputPersons.Rows.Count);
foreach (DataRow person in inputPersons.Rows)
{
    int id = Convert.ToInt32(person[colInputPersons_PERSONID]);
    int homeZone = Convert.ToInt32(person[colInputPersons_HomeZone]);

    //Put new row of person data into person data dictionary
    this.UtilityFunctionParameters.UpdateDecisionMakerVariables(person);

    //if on new zone, get new zonal data and put into zonal alternatives
    if (homeZone != currentHomeZone)
    {
        UpdateVariablesWithMatrices(homeZone, zonalAlternatives);

        currentHomeZone = homeZone;
    }

    //Get new mode choice logsums and put into zonal alternatives
    modeChoiceLogsum = fullCarLogsum.Solve(modeChoiceLogsum);

    for (int index = 0; index < zonalAlternatives.Length; index++)
    {
        zonalAlternatives[index]["DisaggregateModeChoiceLogsum"].Value = modeChoiceLogsum[index];
    }

    string chosenAlternative = MakeChoice(modelStructure);

    AddPersonToOutputForUpdate(dtOutputPersons, chosenAlternative, person, id, homeZone, noCarLogsum);
    rowsProcessed++;
    if ((rowsProcessed % 10000) == 0)
        log.InfoFormat("Thread (0) Processed (1) rows at (2)", System.Threading.Thread.CurrentThread.Man

log.InfoFormat("Person loop completed = (0)", DateTime.Now);
```
Replicability

• By June 1, 2009
  • Developed effective template discrete choice component
  • Developed full-detail database structure
• By November 1, 2009
  • Functional versions of all 27 discrete choice components
  • “Non-hilarious” results for all of them
• After tour mode choice, tour time of day (a tricky one!) took three weeks.
Scalability (a few examples)

• Nowhere in the code do the number of people and households appear.
• Automatic detection of available processors:

```java
// we'll loop over a selection of these by some column
protected DatabaseReference mainInputReference;

protected int minInputID;
protected int maxInputID;

public override Status Run()
{
    Status = Status.OK;
    ReadData();
    if (!nProcessorsToUse.HasValue) nProcessorsToUse = Environment.ProcessorCount;
    int nProcessors = nProcessorsToUse.Value;
    int rows = maxInputID - minInputID + 1;
    int chunkSize = (int)Math.Ceiling((decimal)rows) / (decimal)nProcessors - NUMBER_OF_CHUNKS_PER_PROC
    for (int step = 1; step <= Number0fSteps; step++)
```
Transferrable

• No non-transportation applications yet, but:
• Software calls a diverse set of components:
  • Two simply run TransCAD
  • One processes *PopSyn* outputs
  • Two random distribution simulators
  • One calculates disaggregate logsums
  • One calculates size sum variables
  • 15 execute logit choice
• Currently running on two servers:

Server 1 (4 processors)  
Code Execution

Server 2 (4 processors)  
Database storage and queries

• Various tests on various hardware
Distributable

Suzanne’s or Jen’s desktop (2 processors)  Database Server (4 processors)

Code Execution  Database storage and queries

Jerry’s desktop (4 processors)  Suzanne’s or Jen’s desktop (2 processors)

all operations  All operations
• “Round robin” threading execution upgrade
  • Upgrade Class ModelComponentWithThreads
  • Change how the threads are queued

• Pass-by-reference copy method for decision-agent-specific variables:
  • Upgrade Copy method in Class DecisionMakerSpecificVariable
  • Big run time improvement
Integratable

• Point: to enable seamless integration with other systems at DRCOG

• No examples yet, but plans include:
  • Search model input and out through DRCOG website
  • Integrate with DRCOG regional data model
Tunable

• Number of threads
• “Chunks” per thread

```java
public override Status Run()
{
    Status = Status.OK;
    ReadData();
    if (!nProcessorsToUse.HasValue) nProcessorsToUse = Environment.ProcessorCount;
    int nProcessors = nProcessorsToUse.Value;
    int rows = maxInputID - minInputID + 1;
    int chunkSize = (int)Math.Ceiling((decimal)rows) / (decimal)(nProcessors * NUMBER_OF_CHUNKS_PER_PRO)
    for (int step = 1; step <= NumberOfSteps; step++)
```
Caveman version:

- Relational database goooood!
- OO language goooood!
- Erik still not clear on final runtimes!
- DRCOG tribe learn model reeeeeeel good!
- Learning process hurt head sometimes!
- We learn to fight by standing in middle of battle!
Possible Enhancement

• Automate point-based land use.
• Upgrade of the scenario management system
• Location choice set generation and shadow pricing.
• Integration with DTA
• Enhanced toll modeling
• Upgrades with new survey data
• Model rollout and distribution enhancements.
Jennifer Malm: jen_s_malm@yahoo.com
Suzanne Childress: schildress@drcog.org
Erik Sabina: esabina@drcog.org
DeVon Farago: devonfarago@yahoo.com
Jerry Vaio: gvaio@camsys.com
Scott Meeks: smeeks@camsys.com