SINFERS

A Practical and Modern Rule-Based Expert System

Jason Morris

Morris Technical Solutions LLC

In Partnership with the University of Sydney, Australia

Outline

- **Background Information**
- Knowledge Engineering Aspects
- Using Patterns and Metaphors
- Controlling Rule Execution
- Quick Demo
- Lessons Learned
- Questions & Answers

What is SINFERS?

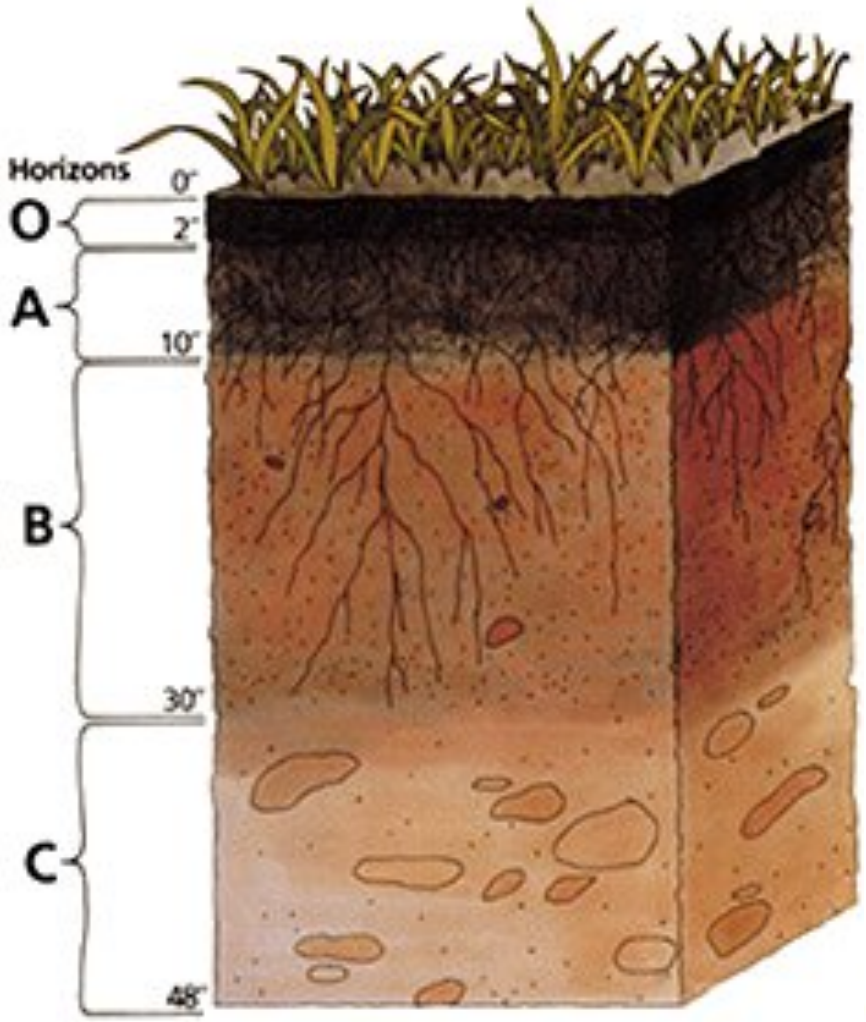
- **S**oil **Infer**encing **S**ystem*
- Based on **pedotransfer functions** (PTFs).
- **Predicts** new soil properties from **existing** properties by applying PTF rules to known soil properties.
- ~ 500 rules w/ ~450 unique

McBratney, A.B., Minasny, B., Cattle, S.R., Vervoort, R.W., 2002. "From pedotransfer functions to soil inference systems", Geoderma 109, pp.41-73.

* http://en.wikipedia.org/wiki/Soil_inference_system

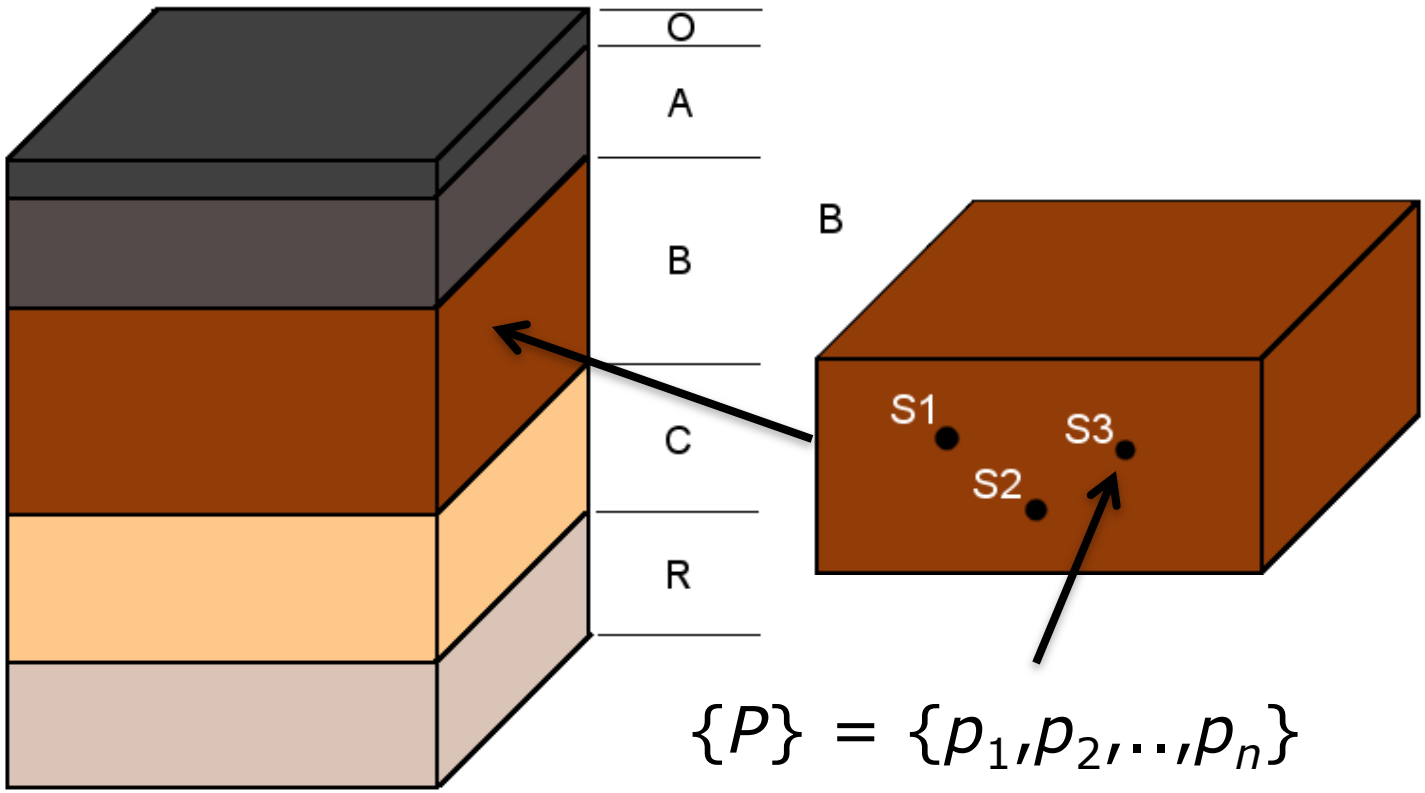
Typical Soil Profile

- Profile composed of layers called **horizons**.
- Horizons have **samples**.
- Samples have **properties**.



Sample Properties

SINFERS estimates soil properties at the sample locations within a horizon.



Pedotransfer Functions Are…

- "*…**translating data we have into what we need***."*
- Linear, multivariate, statistical **regression** functions.
- Derived from **experimental** sample data.
- Used when direct field measurements are **impractical** due to cost, danger, accessibility, or other issues.

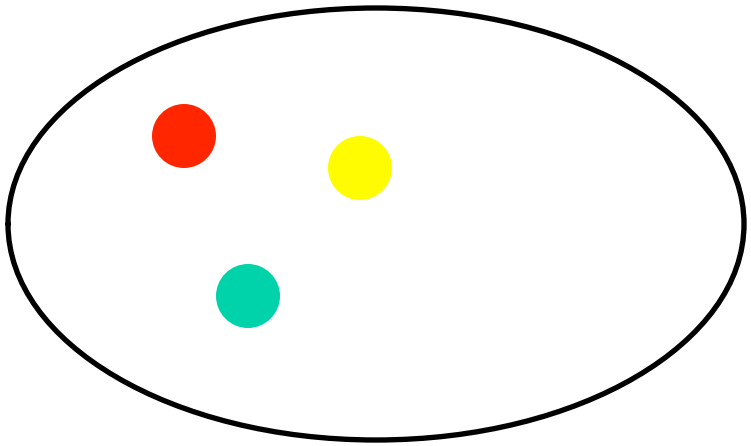
* Bouma, J., "Using soil survey data for quantitative land evaluation", 1989, *Advances in Soil Science* 9: 177–213

Sample PTFs

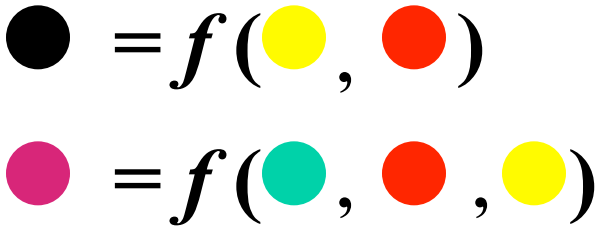
bulk density = f (depth, %_sand)		
P3A1	=	bulk density (g/cm^3)
depth	=	sample depth (cm)
P10_NR_S	=	% sand by weight

```
P3A1 = 1.369 - 0.138035 * depth -
0.321710 * log10(_6A1) + 0.001989 *
P10_NR_S + (P10_NR_S -48.822403)*
(P10_NR_S - 48.822403) * (-0.000088)
```

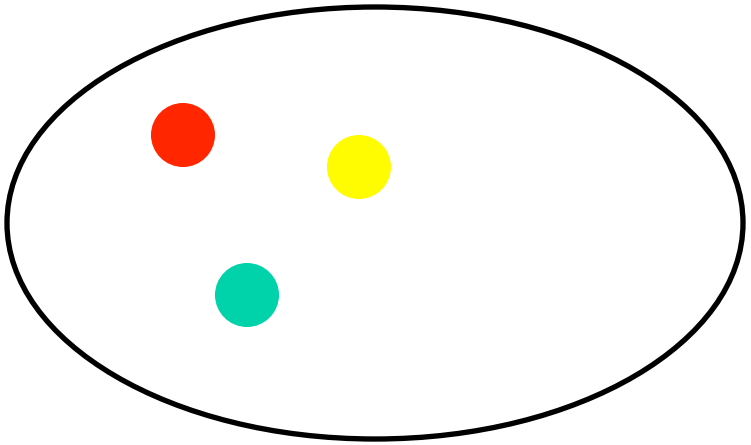
How SINFERS Works



We know that:

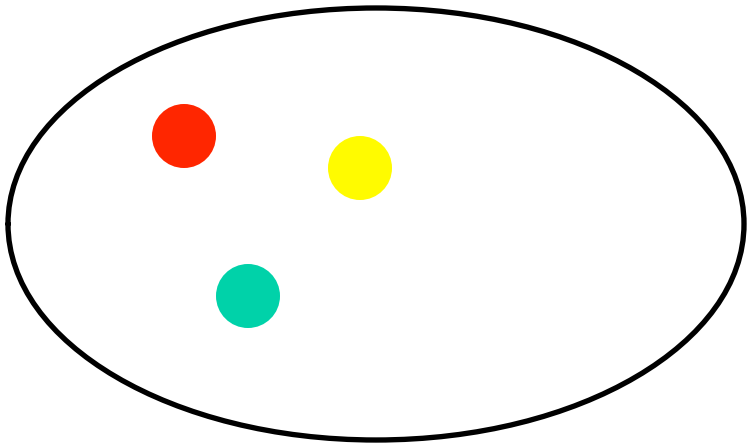


Given
Initial Soil
Properties



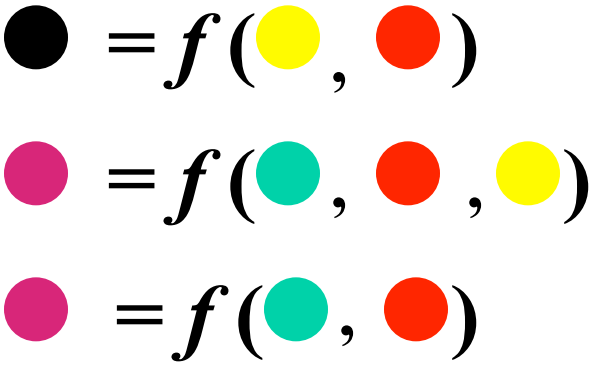
Inferred Soil
Properties

The PTF Problem I



Given
Initial Soil
Properties

What if we have:



We always trust the computed property with the **least** uncertainty regardless of the mean value.

The PTF Problem II

- Each **computed** value has a **mean** value and an **error** associated with it.
- PTFs cannot be used "as is" due to **measurement error** (uncertainty).
- There are many statistical techniques for computing the uncertainty of any given PTF.
- SINFERS uses one based on **Latin Hypercubes** and **Mahalanobis Distance** from centroids.

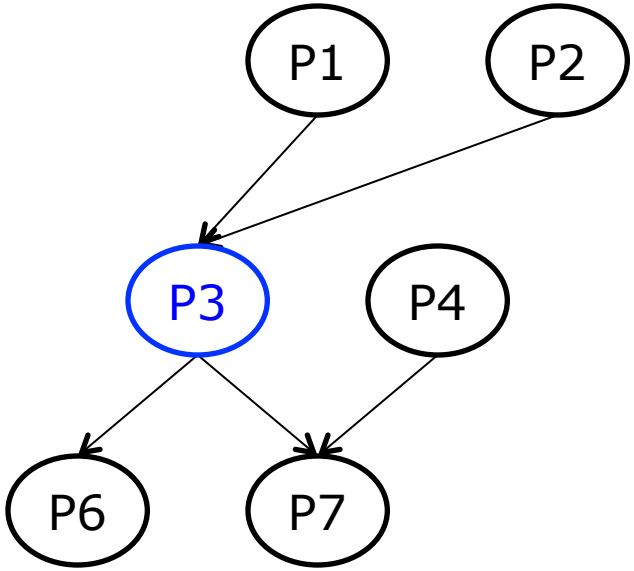
The PTF Problem III

- Users **must** be able to enter new PTF definitions to augment the SINFERS knowledgebase.
- Users are most comfortable entering **free-form** algebraic expressions.
- We **do not** want to hardcode PTF functions into Java methods.
- Yet, we need the PTF expressions to remain **editable** and **computable**.

How do we achieve this???

The Inbreeding Problem I

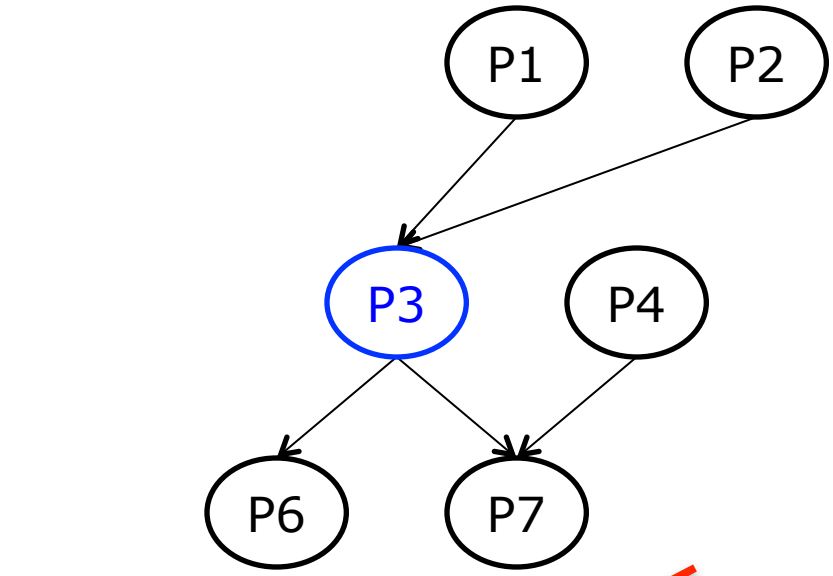
- Let's say SINFERS has an opportunity to re-compute soil property P3 using PTF2.
- However, whereas P3 was originally computed with the original properties P1 and P2, the second iteration of P3 produced by PTF2 has P1, P2, and P7 as arguments.



$$P3 = PTF2\ (P_1, P_2, P_7)$$

The Inbreeding Problem II

- By examining the history for each of the arguments to the candidate PTF2, we see that in the history to argument P7 there is a reference to P3.
- We would be re-computing P3 from an ancestral version of P3, so...



Dependency History

- Each soil property has a set of **functional** dependencies.
- More certain properties are allowed to replace less certain ones.
- We do not allow **circular** dependencies between properties.
- We needed a **simple** mechanism to check this constraint at runtime.

Dependency Checking

Use recursion to do a depth-first search of the dependencies.

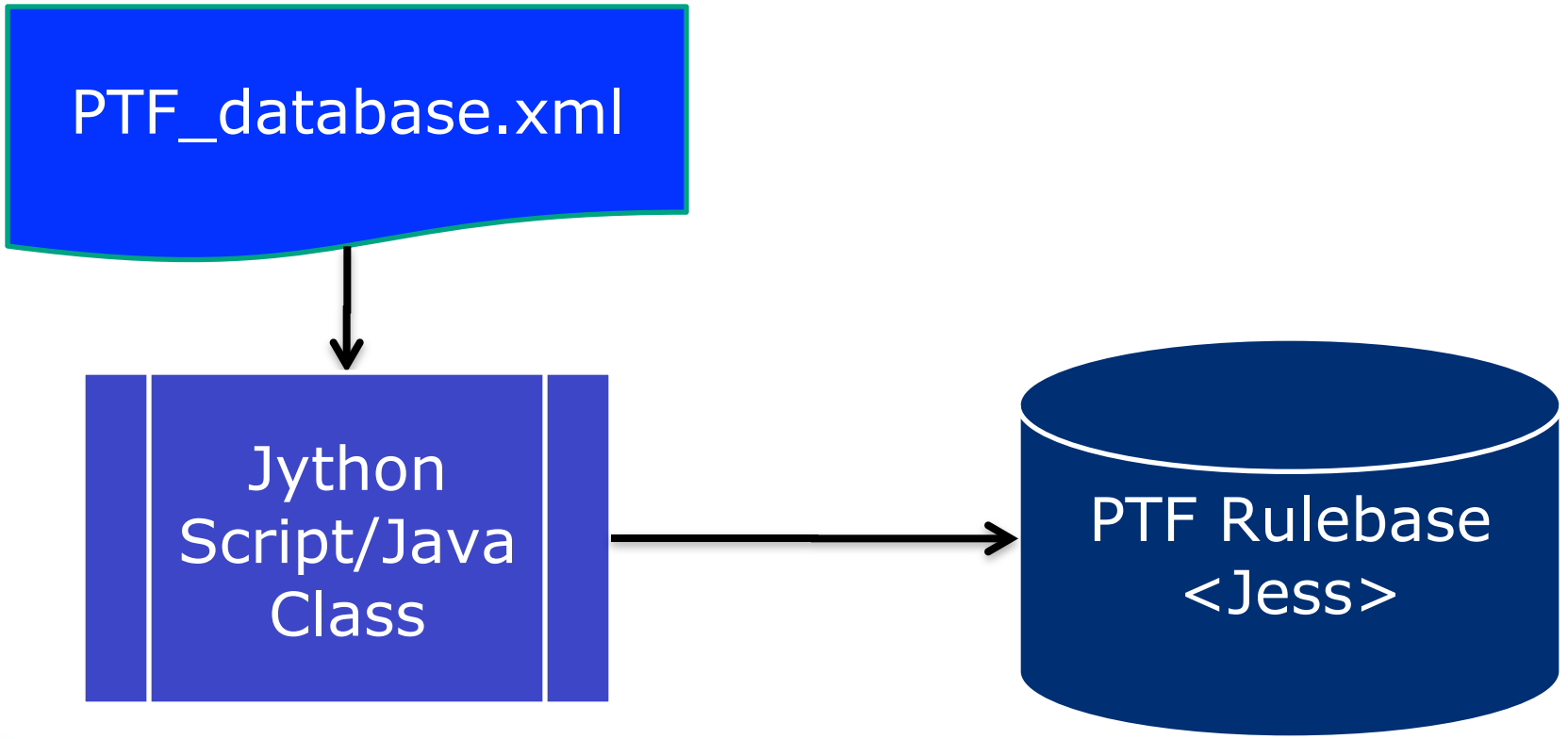
```
public boolean isInDependencyHistory(String lc)
{
  for (SoilProperty p : dependencies) {
  // See if property's lab code matches input
    if (p.getLabCode().equals(lc)) {
      return true;
    } else {
      p.isInDependencyHistory(lc);
    }
  }
 // Fall through if nothing is found
 return false;
}
```

Automation

- In any non-trivial development project, **automation** is key.
- Look for any (boring!?) **repetitive** tasks that can be automated.
- **Scripting** can be a powerful tool for automatically generating code (e.g. Python/Jython, Perl, etc.).
- Saves **time** and reduces **errors**.

Automation Example

Changes to PTF database cause PTF rulebase to be re-built automatically.



Outline

- Background Information
- **Knowledge Engineering Aspects**
- Using Patterns and Metaphors
- Controlling Rule Execution
- Quick Demo
- Lessons Learned
- Questions & Answers

How to Speak Australian

- **Mate** – Friend, buddy, pal.
- **G'Day** – Good morning.
- **No Worries** – It's OK, no problem.
- **Buggered** – All screwed up.
- **Gone Pear Shaped** – *See buggered*.
- **Bob's Your Uncle!** – An epiphany exclamation.
- **Sorted Out** – Fixed, corrected.
- **Dog's Breakfast** – A total mess.

Sample Usage

JASON: Hey, g'day mate!
GRANT: Yeah… g'day!
JASON: Get that algorithm sorted out last night?
GRANT: Nah mate! It was totally buggered.
JASON: A real dog's breakfast, eh? Need a hand?
GRANT: No worries, mate…I've got it sorted out now.

A More Serious Example

From one of our earliest online discussions…

JASON: So let me get this right, we are going to analyze dirt samples and…

GRANT: Nah mate, *dirt* is soil that get's under your finger nails. We're working with *soil*.

Domain Conceptualizing

- Every **domain** has its own vocabulary, jargon, idioms, and expressions.
- Seek first to **understand**, then be **understood** (S. Covey).
- Elicit any and all domain **concepts**, their **properties,** and their **relations** to formulate a **domain ontology**.
- Use the resulting ontology to **shape** and **structure** your rules and algorithms. Refine as needed.

SINFERS Ontology

- Simplified version of the **Soil Information Transfer & Evaluation System** (SITES) Entity Relationship Diagram.
- Interoperability with this standard was a fixed design parameter.
- Resulted in an XML Schema/DTD for marshaling soil sample data.

Outline

- Background Information
- Knowledge Engineering Aspects
- **Using Patterns and Metaphors**
- Controlling Rule Execution
- Quick Demo
- Lessons Learned
- Questions & Answers

Using Design Metaphors

- Creates a **common** understanding of the way forward.
- Helps you **conceptualize** problem and solution spaces.
- Facilitates reasoning by **analogy** which can suggest solution processes.
- Supports spiral/iterative design via **disposable prototypes**.

Design Considerations

- How do we program in the domain vocabulary and not the low-level implementation vocabulary?
- How do we share a conceptualization of managing the inference process?
- How do we anticipate using this API in a web application?

Rule Engine Manager

- REM is like a **manual transmission** for SINFERS.
- **Delegates** to a Rule Manager and a Working Memory Manager.
- Facilitates **asynchronous** communication with the rule engine.

Most importantly, metaphors like this let us program in the domain vocabulary.

Working Memory Manager

- WMM provides management of all high-level interaction between SINFERS and Jess's working memory.
- REM delegates working memory-related tasks to its WMM, and the WMM reports back to its REM when it is finished.

EXAMPLE: To add a candidate soil property to WM

```
addCandidateSoilProperty
(this.ruleEngineManager.getEngine(),
property)
```

Rule Manager

- Provides management of all **high-level** interaction between SINFERS and its various **knowledge-bases**.
- REM **delegates** all tasks involving the loading and unloading of **rules** and the switching of **modules** to the RM class.

Design Considerations

- How do we allow for **multiple processes** for computing PTF values and uncertainties?
- How do we decouple **tools** from their higher-level **processes** so that they can **vary** independently?
- How do we get around the PTF **storage** and **usability** problem?

Algorithms & Evaluators

Concrete Algorithm

Concrete Evaluator A

Concrete Evaluator B

The *Algorithm* Interface

- Specifies the contract of objects that model SINFERS **algorithms**.
- Specifies the **procedure** by which a PTF is computed, but not the **tool** used.
- Delegates the specific **parsing** and **evaluation** functions to its associated *Evaluator* service.
- Allows a PTF to be computed by one or more algorithms (GoF *Strategy* design pattern).

METAL Algorithm

- **M**ean **E**valuation with **T**ot**AL** uncertainty.
- Developed by Grant Tranter, U of Sydney.
- Rapidly computes a PTF mean value and its error.
- Based on FORTRAN routines written by B. Minasny at USyd.

The *Evaluator* Interface

- Specifies the contract of objects that evaluate **expressions**.
- Provides *mechanism* (tool) by which expressions are **parsed** and numerically **evaluated**.
- Separates **responsibility** of managing the high-level computation from the low-level parsing and evaluation.
- SINFERS developers not restricted from using other eval libs in the future.

JEP Parser

- **J**ava (Math) **E**xpression **P**arser.
- Java library for parsing and evaluating mathematical **expressions**.
- Enter an arbitrary formula as a string and instantly **evaluate** it.
- SINFERS currently uses a *JepEvaluator* by default.

ProcessStrategy Interface

Allows us to **vary** how we choose to **process** each element in a **profile**.

sinfers.processing
Interface ProcessStrategy
All Known Implementing Classes:
DefaultHorizonProcessor, DefaultProjectProcessor, DefaultSiteProcessor, DefaultSoilProfileProcessor, DefaultSoilPropertyProcessor, DefaultSoilSampleProcessor

```
public interface ProcessStrategy {
      public void execute(Object target,
Object callbackObj);
}
```

Outline

- Background Information
- Knowledge Engineering Aspects
- Using Patterns and Metaphors
- **Controlling Rule Execution**
- Quick Demo
- Lessons Learned
- Questions & Answers

Problem Solving Methods

- **Propose & Revise**: Suggest solutions and revise them as new information is made available.
- SINFERS uses a modified form of classic Propose & Revise method.
- See Chandraskaran, McDermott, Fensel for papers on generic tasks and problem-solving methods.

Classic Propose and Revise

- **Propose Phase**: A model of an artifact is extended by applying relevant domain knowledge.
- **Revise Phase**: Corrects errors in each extension.
- If a **method** can assume that at each **step** of the process there is always **at most** one applicable extension operator, then no **additional** domain knowledge is required.
- Otherwise, **operator selection knowledge** is needed to discriminate between multiple applicable operators.
- Domain knowledge must include **procedures** and **fixes**.

SINFERS Control Strategy

- **Challenger** vs. **Incumbent**.
- Analogous to political scenario.
- Initially computed value is the **incumbent**.
- Subsequent computations **challenge** the incumbent.
- Challengers are **disqualified** if they violate certain constraints on **circular dependency** and **uncertainty**.

Incumbent vs. Challenger

- Brainchild of the current political climate at the time.
- Idea: Proposed soil properties **challenge** an **incumbent** for the top spot (least uncertainty) in WM.
- Keep rules from doing too much at once. **Many** rules doing **small** changes at a time.

Propose Module

- Use PTF rules to simply **propose candidate** soil properties for consideration as final report-quality values.
- Rules are **manually** focused.
- Focus needs to be manually placed on this module by the RuleEngineManager to start inferencing process. Think of it as putting SINFERS "in gear".

Proposing Rule Example

IF

There is not an existing soil property X,
AND

There is a PTF_X for X, AND

There are soil properties that are the arguments to PTF_X

THEN

Let PTF_X compute X.

Propose that PTF_X define X.

Select Module

- Module is **auto-focus** enabled.
- Automatically switches focus to itself when there are **multiple candidate** soil properties for the **same** lab code in working memory.
- Chooses the **best** soil property facts from among the available **candidates** and promotes winner to **computed** status.

Selecting Rule Example

IF

There is a candidate soil property, p1, for some lab code with id_1, value v_1, and uncertainty u_1, AND

There is not another candidate soil property, p2, for the same lab code with id_2 not equal to id_1, value v_2, and uncertainty u_2 for which $u_2 < u_1$

THEN

Select candidate soil property p1.

Select Rule Example

```
(defrule SELECT::choose-best-contender-for-_14000
"Eliminates weakest contenders for soil property
_14000"
(declare (auto-focus TRUE))
?sp1<-(MAIN::SoilProperty (labCode "_14000")
   (state =(SoilPropertyStateType.INCUMBENT))
   (uncertainty ?e1)(value ?v1)(OBJECT ?objSp1))

?sp2<-(MAIN::SoilProperty
   (labCode "_14000")
   (state =(SoilPropertyStateType.CANDIDATE))
   (uncertainty ?e2&:(< ?e2 ?e1))(value ?v2)
   (OBJECT ?objSp2))
=>
;Remove loser from working memory.
(SELECT::select-action ?sp1 ?sp2 ?objSp1 ?objSp2))
```

Monitor Module

- **Notices** patterns in input, output, and execution behavior that would indicate an **anomaly**, **malfunction**, or **invalid state** in SINFERS.
- **Automatically** switches focus to itself when such events occur.
- Either **corrects** SINFERS's state or **notifies** user if it cannot.

Maintenance Module

- **Cleans up** working memory and handles all in-between chores/ functions when all inferencing has been completed.
- Rules are **manually** focused.
- **Lowest** priority rules of all. Only fire when all else is finished on each analysis run.

Outline

- Background Information
- Knowledge Engineering Aspects
- Using Patterns and Metaphors
- Controlling Rule Execution
- **Quick Demo**
- Lessons Learned
- Questions & Answers

Outline

- Background Information
- Knowledge Engineering Aspects
- Using Patterns and Metaphors
- Controlling Rule Execution
- Quick Demo
- **Lessons Learned**
- Questions & Answers

Lessons Learned

- Follow your passion!
- Seek to understand, then be understood.
- Automate! Automate!! Automate!!!
- Use patterns and metaphors often!
- Know the literature & don't re-invent the wheel.

If this talk has helped a bit, then hopefully your next expert system won't look like…

… A Dog's Breakfast!!!

Thank you all for attending! - JM