Epiphanies of patterns!

A hands-on survey of pattern-related automation technologies
Introductions

• Scott E. Schneider
  – IBM™ Rational™, Pattern Technologies

• Randy Lexvold
  – IBM™ Rational™, Worldwide SME for SOA
Overview

Part I
Patterns
Background

Part II
Hands-On
Survey

Part III
Additional
Resources

**Epiphany?**

**epiphany (plural epiphanes)**

1. A manifestation or appearance of a divine or superhuman being.
2. An illuminating discovery often resulting in a personal feeling of elation, awe, or wonder. Sudden realization of great truth.
3. (Christianity) Season or time of the Christian church year from the Epiphany feast day to Mardi Gras (Shrove Tuesday), the day before Ash Wednesday, the start of Lent (See Epiphany).

Part I
Patterns Background (55 minutes)

1. The ubiquity of patterns
2. Towards the very essence of patterns
3. Man versus machine (versus man and machine)

“Microflame Sunflower”
First Place, 2006 Combustion Art Competition
http://www.cssci.org/images/art2006-MellishEtAl-1277x1502.png

Ben Mellish and Fletcher Miller (National Center for Space Exploration Research); Dan Dietrich and Pete Struk (NASA Glenn Research Center); James Tien (Case Western Reserve University)
Pattern Ubiquituousness

• “There are only patterns, patterns on top of patterns, patterns that affect other patterns. Patterns hidden by patterns. Patterns within patterns.”

• “If you watch close, history does nothing but repeat itself. What we call chaos is just patterns we haven’t recognized. What we call random is just patterns we can’t decipher. What we can’t understand we call nonsense. What we can’t read we call gibberish.”


Pictured is a hand-hammered Zildjian Constantinople ride cymbal – are there patterns to the hammer markings that are too complex to grasp?
Comedy – Rule of Three

• Three programming languages arrive to their first day of school nervously scurrying about looking for home room...

• A pattern observed in some successful jokes (and perhaps some unsuccessful jokes)

• Breaking a sequential pattern that is predicted by the listener happens to be funny:
  – Mundane, mundane, outlandish switch-up

The first language “C” asks for a pointer to home room and he then safely dereferences to the class

The second one, “Prolog” gathers some facts and subsequently reasons her way logically to the class

The third one, Java™, finally stops after the late bell rings and exclaims exceptionally, “ClassNotFound”
Music – Turntablist Scratches

- Forward pattern
- Flare pattern
- Prism pattern
  - Forward flare
  - Backward flare
- Stab pattern
- Drag pattern
- Scribble pattern

Scott E. Schneider ("Deejay Knows-Know") performing occurrences of the scratch patterns listed to the right. Equipment used from left-to-right include a Technics 1200 MKII turntable, a Serato Scratch Live control record and audio interface, as well as a Rane 56 TM mixer.
Mathematics: Golden Ratio

• An interesting natural pattern exists that is related to the Fibonacci series: \{ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... \}

• The ratio of \( f(n+1) \) to \( f(n) \) approaches \( \Phi \) as \( n \) goes to \( \infty \)

• \( \Phi \) is the solution to the quadratic equation: \( n^2 + n + 1 = 0 \)
  \( \Phi = n = 1.61803398874989... \)
  \( \Phi^{-1} = \Phi - 1 \)
  \( \Phi^2 = \Phi + 1 \)

In a golden rectangle, the long side \((a + b)\) is \( \Phi \) times the short side \(a\).
Poetry: Clerihew

- Invented by Edmund Clerihew Bentley (July 10, 1875 – March 30, 1956)

- Four line poem with AABB rhyme pattern

- First line ends in a proper noun, the subject of the poem

- A poetry form can be viewed as a poetry pattern

"Ah, the Singleton,
Not rare like a shingleton,
Although intent is noble,
Is it not just a global?"
Architecture: Alexander

• Christopher Alexander is the father of a pattern-based approach for architecture and his writings are quite poetic and holistic with a strong theme of idealism (1977)

• “A Pattern Language” introduces and relates 253 patterns to form a practitioner vocabulary

• Empowers the individual and community to create their own towns, buildings and constructions (a timeless way of building as observed over time)
A decade later, in 1987, Kent Beck and Ward Cunningham applied the idea of an Alexander-inspired pattern language to empower users to construct their own user interfaces and applications.

After building momentum at OOPSLAs, in 1993, Grady Booch and Kent Beck host a Colorado mountain retreat that sparks the formation of the Hillside Group.

In 1994, Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, the collectively known Gang-of-Four released the ever-so-popular “Design Patterns: Elements of Reusable Object-Oriented Software”
Part I
Patterns Background (55 minutes)

1. The ubiquity of patterns
2. Towards the very essence of patterns
3. Man versus machine (versus man and machine)

“Patterned” by Neko-Matt
(Pachinko balls in a sake cup)
What is a Pattern? (General)

- A pattern is a recurring abstraction that specifies a discriminating, context-sensitive configuration of objects and the relationships between them.

- The relationships may be of a sequential, spatial, temporal, logical, mathematical or other characteristically observable or quantifiable nature.

http://karenswhimsy.com/mushrooms.shtm
What is a Pattern? (Alexandrian)

• “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”

Christopher Alexander, “The Timeless Way of Building”
http://karenswhimsy.com/islamic-architecture.shtm
http://www.flickr.com/photos/pingnews/462173771/
Qualities Imparted

• **Regularity**
  – Patterns impart a degree of consistency on the domains that exhibit them

• **Predictability**
  – Patterns offer expectedness, advantages and disadvantages already documented, relationships understood

• **Efficiencies**
  – Patterns offer practitioner productivity gains due to their reusability

• **Communicability**
  – Patterns build a vocabulary for domain practitioners to converse with

• **Habitualness**
  – Patterns become ingrained with the practitioner over time, the glue for the other core domain concepts

• **Maturation**
  – Patterns enable the maturation and evolution of a domain, “standing on the shoulders of giants”
State Pattern

Role = Context

Relation = delegates-to

Role = State

Relation = abstracts

Role = Concrete State
State Occurrence

Participant = Signal assumes the role Context

Participant = Light assumes the role State

Participants = Green, Yellow and Red assume the role Concrete State
State Decomposition

- Patterns can be composites which are comprised of component patterns
- Patterns that cannot be further decomposed into component patterns are commonly referred to as micro-patterns, atomic or elemental patterns
Part I
Patterns Background (55 minutes)

1. The ubiquity of patterns
2. Towards the very essence of patterns
3. Man versus machine (versus man and machine)

“Hubble Looks at Monocerotis”
Light echo reveals dust patterns around star

NASA, ESA, and The Hubble Heritage Team (STScI/AURA)
Operations

Pattern

Apply
• Impose a known pattern on artifacts

Detect
• Discover occurrences of known patterns

Harvest
• Gather and codify new patterns from occurrences

Mine
• Discover recurrences of potential uncharted patterns

Visualize
• Represent patterns or occurrences graphically

...
Use Case MAD LIBS™

- **Operations**
  - Apply, detect, harvest, mine, visualize, ...

- **Domains**
  - Analysis, requirements, architecture, design, implementation, testing, deployment, process, ...

- **Classes**
  - Patterns, anti-patterns, macro-patterns, micro-patterns, choreographic patterns
Manual Execution

Mapping operational use cases to **human** executable tasks:

- **Apply (create occurrences)**
  - A person reads from the available literature and interprets the natural language and pictorial description, and if it is suitable, translates it into a sequence of steps used to apply the pattern to an existing situation

- **Detect (find occurrences)**
  - A person studies existing bodies using observational skills to identify known pattern occurrences, cross-referencing to the literature when necessary

- **Harvest (create patterns)**
  - A person generalizes existing concrete occurrences into an acceptable pattern abstraction and documents it in a standard natural language and pictorial format

- **Mine (find patterns)**
  - A person sifts through existing bodies using instinct and luck to stumble upon new patterns using recurrence observation, after a few occurrences are confirmed, harvesting is possible
Machine Execution

Mapping operational use cases to **machine** executable tasks:

- **Apply (create occurrences)**
  - A machine comprehends a formal pattern specification and derives executable code that transforms the starting body state to one that exhibits the pattern.

- **Detect (find occurrences)**
  - A machine comprehends a formal pattern specification and uses that to form an appropriate query that will execute against existing bodies seeking concrete occurrences of that pattern.

- **Harvest (create patterns)**
  - A machine considers an existing pattern occurrence that is not yet formally specified, analyzes it and generates a formal pattern specification that satisfies the particular pattern occurrence.

- **Mine (find patterns)**
  - A machine uses a particular strategy to traverse existing bodies, using recurrences as hints to existing pattern occurrences that are yet to be formally specified.
Approaches

**Declarative** pattern specifications are separate from the generic operational programs that interpret them

- One can imagine creating a domain-specific language for pattern specification where specific patterns are specified using a language that includes concepts such as patterns, occurrences, roles, participants, structure and behavior
- This is how some pattern automation research is approached in academia, basically solving the more general problem and then applying it to specific solutions
- More burden is on the tool developer and less burden is on the pattern practitioner

**Imperative, executable representations of pattern specifications, are developed specifically for select operational use cases**

- These small programs can be thought of as micro-tools and unlike more general tools these tools are used for very specific purposes (for example, one can create a Java program that knows how to detect a class of Javadoc anti-patterns in code)
- This is how most pattern automation support is implemented in the industry, each new pattern and supported operation is coupled together and is not easily reusable in other patterns or operations
- More burden is on the pattern practitioner and less burden is on the meta-tool developer, the meta-tool is the tool that assists in micro-tool implementation
Man and Machine

It's not the tools that solve problems, it's the people that use the tools that solve problems...

“We are all part of some cosmic pattern, and this pattern works toward good and not evil. It builds and does not destroy.”

*Henry Kuttner, “The Creature from Beyond Infinity” (1940)*

“Art is the imposing of a pattern on experience, and our aesthetic enjoyment is recognition of the pattern.”

*Alfred North Whitehead, “Dialogues” (1954)*

*Works of Fritz Kahn (1888 – 1968)*
Part II
Hands-On Survey (55 minutes)

1. Apply patterns to development artifacts
   - JET/QVT* hands-on lab

2. Visualize and interact with pattern representations
   - EMF/GMF hands-on lab

3. Detect and validate patterns and occurrences*
   - EMFQ/EMFV hands-on lab

“Jupiter”
Ganymede (top moon with shadow)

NASA/Johns Hopkins University Applied Physics Laboratory/Southwest Research Institute
Apply Patterns (JET2/QVT*)

- Walk-through building a pattern application micro-tool with JET2 (using model-to-text transformations to apply patterns to textual artifacts)

- Discussion of using QVT to apply patterns in-place (using model-to-model transformations to apply patterns to model-based artifacts)
Visualize Patterns (EMF/GMF)

- Discussion of using EMF to represent pattern (creating an Ecore model)

- Walk-through building a graphical interface micro-tool with GMF (using graphical front-end to visualize and interact with pattern)
Detect Patterns* (EMFQ/EMFV)

- Discussion of using EMFQ to detect pattern occurrences (creating queries to retrieve pattern occurrences)

- Discussion of using EMFV to validate pattern occurrences (plugging in to the validation framework for problem reporting)
Part III
Additional Resources

1. Eclipse™ based commercial pattern automation tools
2. Other open source pattern-related technologies
3. Interesting research in and around patterns
Offerings: Commercial

- IBM Rational
  - Software Architect
  - Software Modeler
  - Systems Developer
  - Application Developer
  - Data Architect

- Borland™ Together
  - Architect, Designer, Developer

- Compuware™
  - OptimalJ, MagicDraw

Hearn's, New York City, New York, Pattern Department, September 22, 1944
Offerings: Non-Commercial

• Concepts that coalesce with pattern automations:
  – Model-driven engineering ("MDE")
  – Pattern-driven engineering ("PDE")
  – Domain specific languages and modeling ("DSL", "DSML")
  – Artifact transformations ("M2M", M2T", “T2M”, “MoP”)
  – Query and search, graph theory, constraint satisfaction problems ("CSP")
  – Program understanding, static and dynamic analysis
  – Model representation and synchronization

• Search the internet using these concepts as a start to find Eclipse-based and other open source offerings
Research

• **PROSE** (“Pattern Reasoning and Object-Oriented Specification Environment”)
  – Scott E. Schneider, IBM Rational

• **SPQR**, (“System for Pattern Query and Recognition”)
  – Jason McC. Smith, IBM Watson

• **MDCE** (“Model-Driven Constraint Engineering”)
  • Michael Wahler, Jana Koehler, Achim D. Brucker, IBM Zurich

• **Model Grokking**
  • Dany Moshkovich, Shiri Kremer-Davidson, IBM Haifa
Thanks!

http://flickr.com/photos/uhuruj701/2252905347/