Comparing Software Abstractions

Baby Steps

Michael Hansen

Lab Lunch Talk 2011

INDIANA UNIVERSITY
PERVASIVE TECHNOLOGY INSTITUTE

Comparing Abstractions

- Need objective comparison method
  - Libraries (OpenGL vs. Direct3D)
  - Language constructs (λ-expressions, concepts in C++)
- Motivation
  - Inform evolution of libraries, languages
  - Widen audience
  - Education

Proposal

- Relative to complexity metrics
- ***Abstractions should decrease "complexity"***
- Which metrics?
- Whose complexity?

$$A_P = \sum_{|\cdot| \in C} w_{|\cdot|}(|P| - |P'|)$$

Yet Another Problem

- Need reasonable complexity metrics
  - ***Weyuker's*** Properties
- Some classics
  - Statement Count
  - McCabe Cyclomatic Number
  - Halstead Effort Measure
  - Oviedo Data Flow
- A few newbies
  - ***"Cognitive Complexity"***
  - Kolmogorov Complexity
  - Veldhuizen metrics
  - Chunking

Weyuker's Properties (1/3)

- P, Q, R: program bodies
  - All free variables assigned default values
- P; Q: P and Q concatenated
- |P|: complexity of P, c(P)
- ***P ≡ Q: P and Q are functionally equivalent***
  - Halt on same inputs, produce same output

Weyuker's Properties (2/3)

1. $(\exists P, Q)(|P| \neq |Q|)$ — Not all same complexity
2. $(\forall c)(\{P \mid |P| = c\} \text{ is finite})$ — Finite # of programs of a given complexity
3. $(\exists P, Q)(|P| = |Q| \text{ and } P \neq Q)$ — Not all different complexities
4. $(\exists P, Q)(P \equiv Q \text{ and } |P| \neq |Q|)$ — Functional equivalence != complexity equivalence

Weyuker's Properties (3/3)

5. $(\forall P, Q)(|P| \leq |P; Q| \text{ and } |Q| \leq |P; Q|)$; does not decrease $|\cdot|$
6. $(\exists P, Q, R)(|P| = |Q| \text{ and } |P; R| \neq |Q; R|)$ Context matters
7. $(\exists P)(|P| \neq |\text{permute}(P)|)$ Order matters
8. $(\forall P)(|P| = |\text{rename}(P)|)$ Identifier names do not matter
9. $(\exists P, Q)(|P| + |Q| < |P; Q|)$ Gestalt programs

Statement Count

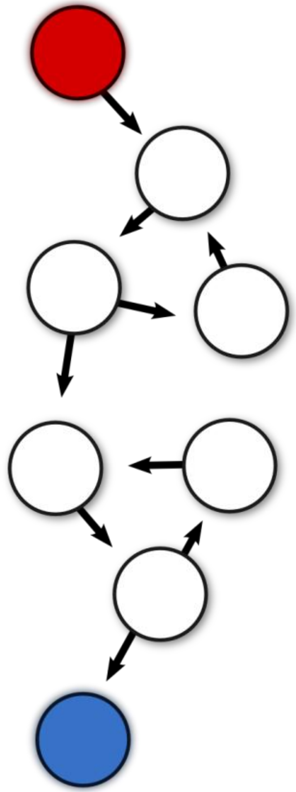
- ***Need definition of "statement"***
  - Physical source code line?
  - Logical source code line?
- Easy to compute!
- Correlated with defects, other metrics
  - Executable lines of code
  - 15-20 bugs per KLOC?

Statement Count

- Property 6 – context matters
  - Fixed for a given block
- Property 7 – order matters
  - Line order is irrelevant
- Property 9 – whole may be greater than sum of parts
  - Consequence of 6

Cyclomatic Complexity (McCabe, 1976)

- Count of linearly-independent paths
- Edges – Nodes + 2 * Connected Components
  - 9 – 8 + 2 * 1 = 3
- Split modules if CC > 10
- ***Branch Coverage ≤ CC ≤ Paths***
  - Upper-bound on test case branch coverage
  - Lower-bound on paths through control flow graph



Cyclomatic Complexity

- Property 2 – only finite # of programs have a given comp.
  - Only decision structure matters
- Property 6 – context matters
- Property 7 – order matters
- Property 9 – whole may be greater than sum of parts

Effort Measure (Halstead, 1977)

- n_1 = distinct operators, n_2 = distinct operands
- N_1 = all operators, N_2 = all operands
- Measures
  - Program length: $N = N_1 + N_2$
  - Program vocabulary: $n = n_1 + n_2$
  - Volume: $V = N * \log_2(n)$
  - Difficulty: $D = (n_1 * N_2) / (2 * n_2)$
  - Effort = $D * V$

Effort Measures

- Property 5 – concatenation cannot decrease comp.
  - Overlap in operators
- Property 7 – order matters
  - Only counting operators, operands

Data Flow Complexity (Oviedo, 1980)

- Program is broken into blocks
  - Statements executed as a unit
- Path from block A to block B
  - Control flow from A to B (i.e. GOTO)
- Variable reaching block B
  - Defined in previous block
  - Not redefined in path (including B)

$$DF_i = \sum_{j=1}^{\|V_i\|} \text{definitions}(v_i)$$

$$DF = \sum_{i=1}^{\|S\|} DF_i$$

Data Flow Complexity

- Property 2 – only finite # of programs have a given comp.
  - Block size is irrelevant
- Property 5 – concatenation cannot decrease comp.
  - Only interblock data flow is considered

Kolmogorov Complexity

- Easy to define, hard to compute

$$|P| = (\min_{l} \mid l = \text{length}(Q) \wedge P \equiv Q)$$

- Property 1 – not all the same complexity
- Property 2 – only finite # of programs have a given comp.
- Property 3 – not all different complexities
- Property 4 – functional equiv. != complexity equiv.

Kolmogorov Complexity

- Property 5 – concatenation cannot decrease comp.
  - Repeated blocks are compressed
- Property 6 – context matters
  - Non-functional code may be used
- Property 7 – order matters
  - Different function
- Property 9 – whole may be greater than sum of parts
  - Concatenation can only decrease complexity

Cognitive Complexity (OO)

- ***Based on "Cognitive Informatics"!***
- Each class method assigned weight
  - Sequence = 1, branch = 2, iteration = 3, call = 2
- Class weights
  - Added for same level
  - Multiplied for different levels (parent, child)
- Correlated with class coupling

Cognitive Complexity (OO)

- Property 6 – context matters
  - Weights are fixed for a class
- Property 7 – order matters
  - Method order is irrelevant
  - What about inside methods?

Metrics and *Weyuker's* Properties

	Lines	Cyclomatic	Effort	Data Flow	Kolmogorov	Cognitive
1						
2						
3						
4						
5						
6						
7						
8						
9						

Veldhuizen Metrics

- Token count (Minimum Description Length)
  - Related to Kolmogorov Complexity
  - Best = min x | x = model tokens + instance tokens
- Inversion difficulty
  - Locate suitable abstraction, parameters
  - Substitution – unification
  - Common inversions are low computational complexity

Chunking (Cant et al 1995)

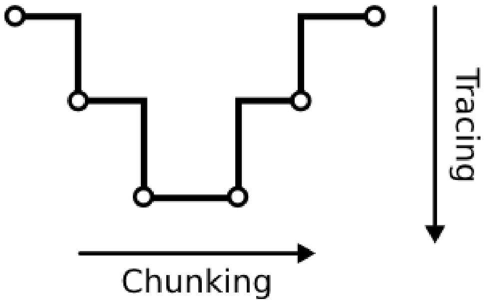
- Short-term memory
  - 7 ± ***2 "chunks"***
  - Capacity expanded by chunking
  - Distraction = forgetting after 20-30 sec.
- Long-term memory
  - Virtually unlimited capacity
  - Structure, low noise enhance recall
  - Chunking and LTM structure are related

Chunking

- Variable plan
  - Variables have roles (iterators, user input, etc.)
  - Names are crucial, even for experts
- Control flow plan
  - Common control flow structures
  - Syntactic representation important
    - ***“while” instead of “if”***

Chunking

$$C_i = R_i + \sum_{j \in N} C_j + \sum_{j \in N} T_j$$



- ***Program is broken into N "chunks"***
  - Decision or loop structure
- C_i = complexity of i-th chunk
- R_i = difficulty of understanding
- T_i = difficulty of tracing dependencies

Chunking

$$C_i = R_i + \sum_{j \in N} C_j + \sum_{j \in N} T_j$$

$$R_F(R_S + R_C + R_E + R_R + R_V + R_D)$$

Famil. (Size + Ctrl Struct. + Bool Expr. + Recog. + Visual Struct. + Disrupt.)

Chunking

$$C_i = R_i + \sum_{j \in N} C_j + \sum_{j \in N} T_j$$

$$T_F(T_L + T_A + T_S + T_C)$$

Famil. (Localization + Ambiguity + Spacial Dist. + Cueing)

Reservations

- Metrics
  - Purely syntactic, uncomputable, vague/subjective
  - Actual cognitive models?
  - All code is rarely available or needed
- Properties
  - Renaming (property 8) – obfuscation
  - Concatenation – really?
  - Independent of programmer

Future Directions

- Metrics relative to
  - Domain/Perspective
    - Tolerance, user, developer
  - Programmer
    - ***Tools matter less than skill, "rules of discourse"***
  - Task
    - Reading, editing, debugging
- Cognitive Dimensions of Notation Framework

Questions?