EE382V

System Design Metrics

Mark McDermott

“You cannot control what you cannot measure”

From *Controlling Software Projects, Management Measurement & Estimation*
by Tom DeMarco

Agenda

- Design metrics
- Complexity Models
- SW Metrics
- HW Metrics
- Process Flow Metrics

Motivation for design metrics

- The questions managers always ask:
  - How risky is this design?
  - How much will it cost to implement?
  - Have we tested enough?
  - Should we reuse or implement from scratch?
  - What are our defect rates?
  - How well are we using our computers and tools

What can we do with Metrics?

- Manage risk
  - Reduce the probability of an issue becoming a problem!
- Manage projects, not by the seat of the pants, but with insight into the performance of the developer
  - An objective, quantitative basis for evaluating product quality and analyzing issues/problems
  - A foundation for quantitative control of the program management and engineering processes.
    - Program performance vs. program plans
    - Cost and schedule control
    - Quality & Configuration Control
    - Defect tracking
    - Staffing
    - Process improvement

"Early" design metrics

- Early availability of metrics is a key factor to a successful management of software and hardware development, since it allows for:
  - Early detection of problems in the artifacts produced in the initial phases of the lifecycle (specification and design documents) and, therefore, reduction of the cost of change - late identification and correction of problems are much more costly than early ones;
  - Better hardware/software quality monitoring from the early phases of the life-cycle;
  - Quantitative comparison of techniques and empirical refinement of the processes to which they are applied;
  - More accurate planning of resource allocation, based upon the predicted error-proneness of the system and its constituent parts.

Definition of Metrics

- Metrics: The collection of activities concerned with measurement in software and hardware engineering.
- A metric is a measure of some aspect of a program, design, or algorithm.
  - It can be systematically calculated
  - It can be used to make inferences about that program, design, or algorithm.
- By systematically calculating values for programs of known complexity - We can infer the complexity of other programs from their calculated values.
  - For example:
    - We know that programs, designs, and algorithms with y value for metric x had problem z
      - High defect rate, poor maintainability, etc.
    - If my program has those metrics
      - It will probably have similar problems

Measurement vs. Metric

- Measures vs. Metrics
  - Measures are the numbers used to create the metrics
  - Metrics are the numbers turned into information
  - Example
    - Measure: A linearly independent path through a module
    - Metric: Cyclomatic Complexity = 25
      - The total number of linearly independent paths through a module
  - Example
    - Measure: Dollars Budgeted
    - Measure: Dollars expended
    - Metric: Cost Performance Index (CPI)

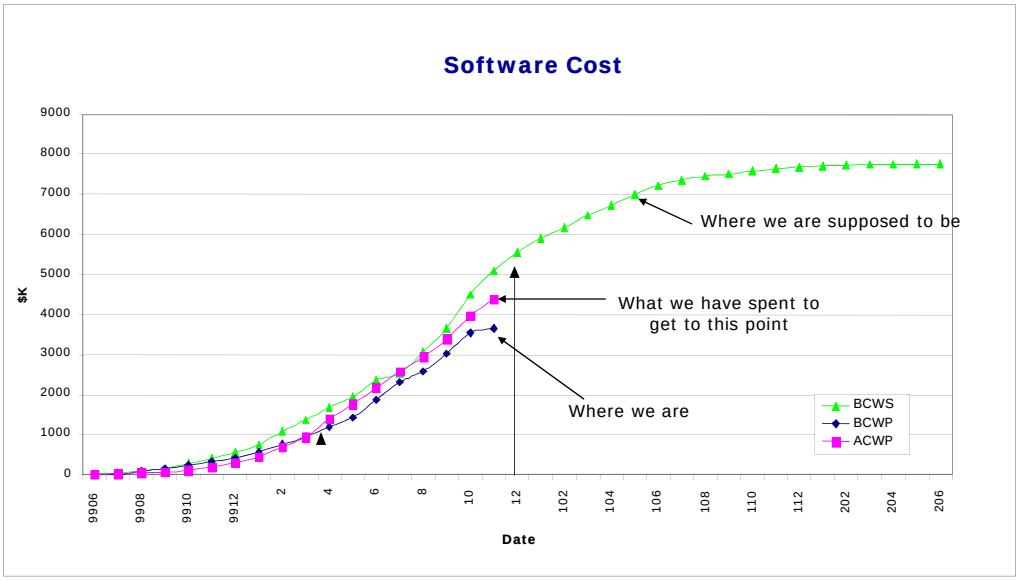
Types of Metrics

- Direct measurement: involves no other attributes or entities
  - Lines of code (LOC)
  - Number of transistors
- Indirect/derived measurement: combination of multiple other measurements
  - Defect density = number of defects/LOC)
- Predictive measurement: use mathematical models—measure known values, interpret results (e.g.: predicting implementation cost)
  - Ex: COCOMO for effort prediction

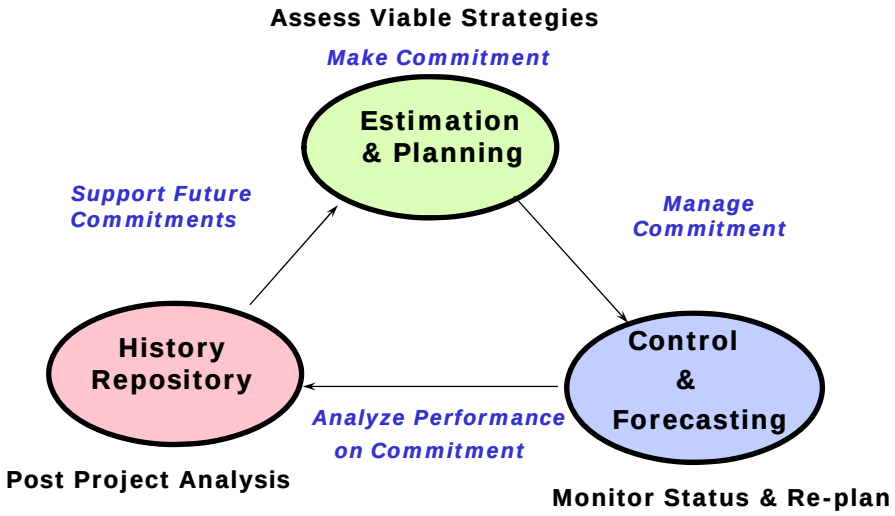
More Examples of Metrics

- Software size: lines of code
- Personnel cost: salary, productivity
- Modularity: function call fan-in and fan-out
- Test adequacy: coverage, detection rate
- Process: defects per KLOC, # of bug reports
- Dependability: reliability, availability, MTBF
- Reuse: percent code reused
- Productivity: LOC per day
- Performance: CPU and memory usage, MIPS, FLOPS

Cost Performance Metric Example



Optimal Project Control Flow



Downside of Metrics

- Management loves metrics as they abstract the enormous amounts data that management needs to make decisions.
- Metrics rarely point at the root cause when measurements don't meet expectations. This is caused by:
  - Wrong types of measurement.
  - Too much abstraction.
  - Too detailed.
  - Lag time of the indicators - resulting in stale data.
- There is a tendency to create centralized organizations to manage the measurement and develop metrics.
  - The focus turns from getting the job done to meeting the numbers.

Agenda

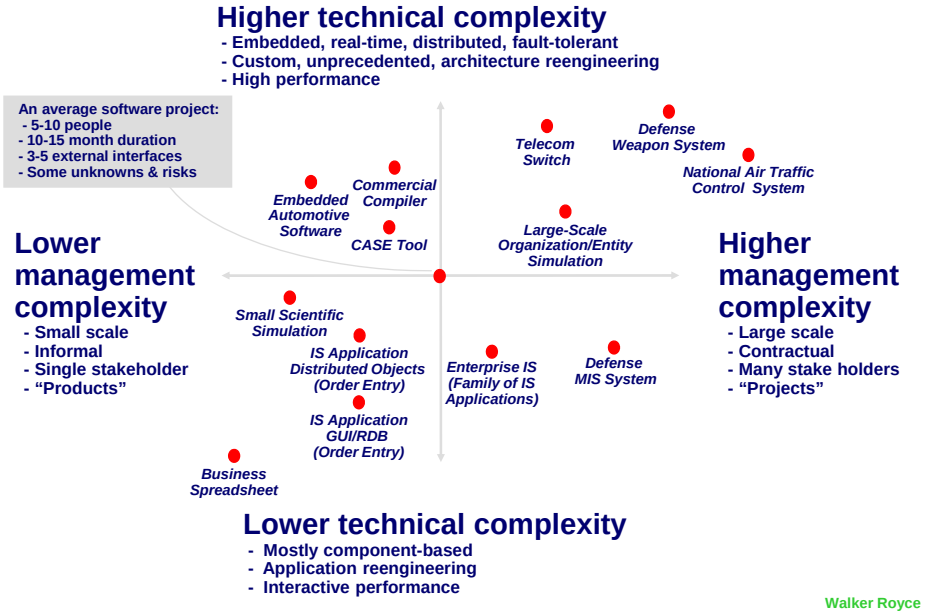
- Design metrics
- Complexity Models
- SW Metrics
- HW Metrics
- Process Flow Metrics

Complexity Models

- In general *reliability is inversely related to complexity*
- Measures of software complexity
  - McCabe
  - Halstead
  - Function Points

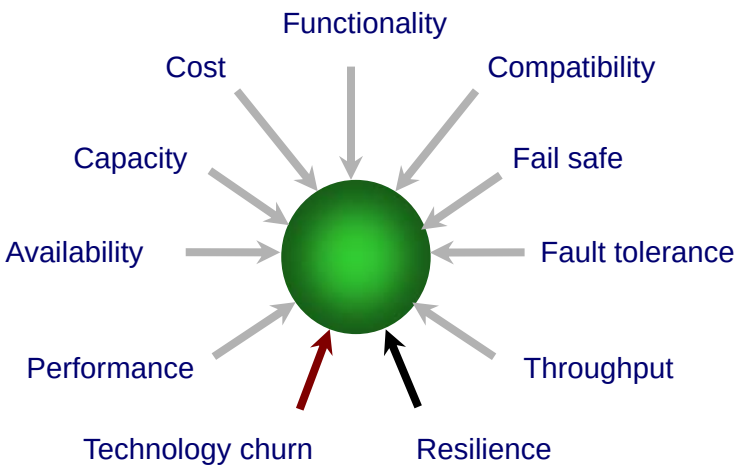
  } Count branches, calls, inputs, outputs etc.
- Measure of hardware complexity
  - Ratio of control transistors to datapath transistors
  - Ratio of high speed I/O signals to slow speed I/O signals
  - Silicon process: gate length, # layers of metal, # Vt's, double patterning requirements.

Dimensions of software complexity



Walker Royce

Complexity Forces



"The challenge over the next 20 years will not be speed or cost or performance; it will be a question of complexity."

Bill Raduchel, Chief Strategy Officer, Sun Microsystems

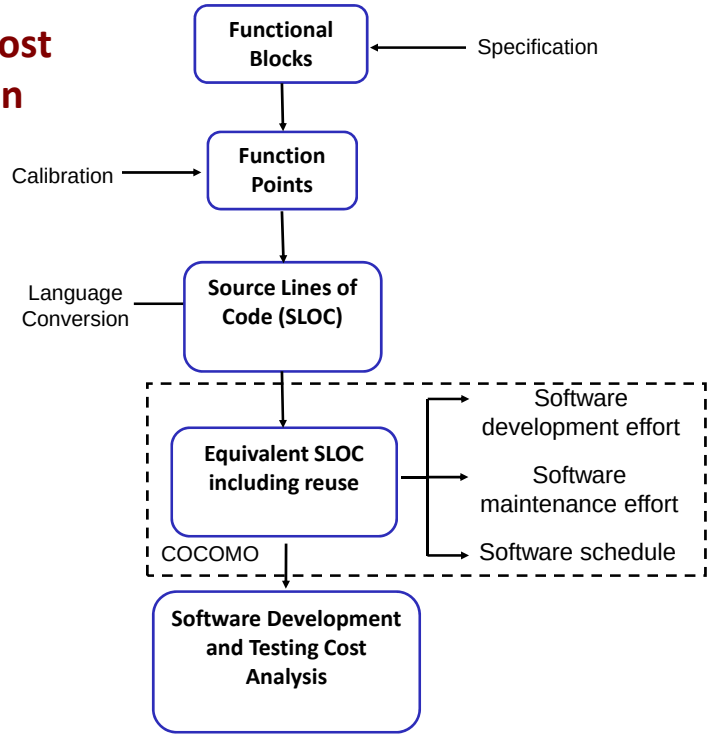
Agenda

- **Design metrics**
- **Complexity Models**
- **SW Metrics**
- **HW Metrics**
- **Process Flow Metrics**

Metrics used for SW Development

- Lines of Code (LOC)
- Function Points (FP)
- FFP (Full Function Points)

Software Cost Estimation Process



Computing Functions Points

- **Number of user inputs**
  - **Distinct input from user**
- **Number of user outputs**
  - **Reports, screens, error messages, etc**
- **Number of user inquiries**
  - **On line input that generates some result**
- **Number of files**
  - **Logical file (database)**
- **Number of external interfaces**
  - **Data files/connections as interface to other systems**

Function point and SLOC metrics

- FP and to some extent SLOC based metrics have been found to be relatively accurate predictors of effort and cost
- Need a baseline of historical information to use them properly
- Language dependent
- Productivity factors: People, problem, process, product, and resources
- FP cannot be reverse engineered from existing systems easily
- Function points can be used as part of a indirect/derived measurement
  - **Errors per FP**
  - **Defects per FP**
  - **Cost per FP**
  - **Pages of documentation per FP**
  - **FP per person month**

FP and Languages

Language	LOC/FP
Assembly	320
C	128
COBOL	106
FORTRAN	106
Pascal	90
C++	64
Ada	53
VB	32
SQL	12

Where does Verilog or VHDL fit in?

Function Points

Classify each component of product (Inp, Out, Inq, Maf, Inf) as simple, average, or complex.

- **Assign appropriate number of function points**
- **Sum gives UFP (unadjusted function points)**

	Level of Complexity		
Component	**Simple**	**Average**	**Complex**
Input item	3	4	6
Output item	4	5	7
Inquiry	3	4	6
Master file	7	10	15
Interface	5	7	10

Function Points (cont)

- **Compute technical complexity factor (TCF)**
  - **Assign value from 0 ("not present") to 5 ("strong influence throughout") to each of 14 factors such as transaction rates, portability**
  - **Add 14 numbers ⇒ total degree of influence (DI)**

    TCF = 0.65 + 0.01 × DI
  - **Technical complexity factor (TCF) lies between 0.65 and 1.35**

1. Data communication
2. Distributed data processing
3. Performance criteria
4. Heavily utilized hardware
5. High transaction rates
6. Online data entry
7. End-user efficiency
8. Online updating
9. Complex computations
10. Reusability
11. Ease of installation
12. Ease of operation
13. Portability
14. Maintainability

Function Points (cont)

FP = Total Count * [0.65 + .01*Σ(Fi)]

Total count is all the counts times a weighting factor that is determined for each organization via empirical data

Fi (i=1 to 14) are complexity adjustment values

Analysis of Function Points

- Function points are usually better than KDSI (Thousands of Delivered Source Instruction)
- As with any model there are always inaccuracy.
  - Errors in excess of 800% counting KDSI, but *only* 200% in counting function points.
- Like FFP, maintenance can be inaccurately measured
  - Never underestimate the maintenance phase. It can cost more than the development phase.

Complexity Adjustment

- Does the system require reliable backup and recovery?
- Are data communications required?
- Are there distributed processing functions?
- Is performance critical?
- Will the system run in an existing heavily utilized operational environment?
- Does the system require on-line data entry?
- Does the online data entry require the input transaction to be built over multiple screens or operations?

Complexity Adjustment (cont)

- Are the master files updated on line?
- Are the inputs, outputs, files, or inquiries complex?
- Is the internal processing complex?
- Is the code designed to be reusable?
- Are conversions and installations included in the design?
- Is the system designed for multiple installations in different organizations?
- Is the application designed to facilitate change and ease of use by the user?

Agenda

- Design metrics
- Complexity Models
- SW Metrics
- HW Metrics
- Process Flow Metrics

HW Development Metrics

- Architectural complexity
- RTL lines of code
- Number of transistors
- Die size, power, schedule
- Type of design: ASIC, Full Custom, FPGA, Platform Based
- Types of circuitry designed: Random logic, Datapath, memory array and analog.
- The frequency of operation including compensation for inherent process speed (gate-delay).
- Amount of reuse: FE (RTL, logic) and BE (physical)
- Process utilization including compensation for process technology.

Useful Figures-of-Merit (FOM)

- Transistor Normalization (T-NORM)
  - The "T-NORM FOM" is generated by using the cost drivers to normalize the transistor counts for a given design. This attempts to account for the fact that not all transistors are created equally ;-)
- Productivity FOM
  - The "Productivity FOM" would be derived by the ratio of the normalized transistor count to the project duration times the productivity cost drivers. The productivity cost drivers attempt to account for the fact that not all designers equally capable and that management has a huge impact.

Generating a T-NORM FOM

- From a technology perspective the interesting cost drivers that impact transistor normalization are:
  1) Types of circuitry designed: Random logic, datapath, memory array and analog.
  2) The frequency of operation including compensation for inherent process speed (gate-delay).
  3) Amount of reuse: FE (RTL, logic) and BE (physical)
  4) Process utilization including compensation for process technology.

Cost drivers and normalization coefficients

1) Types of circuits:

- There are four basic types of circuits:
  - Analog – PLL, Sense Amps, I/O, mixed signal, etc.
  - Random control logic implemented with random logic
  - Memory – LSA & SSA
  - Datapath – custom or structured (CBD) implementation

- Cost drivers for each type:
  - Analog: 3 -10
  - Random Logic: 1-2
  - Datapath: .2 – 1.2
  - Memory: .1 - .8

Note: the smaller the cost (driver) the better the productivity.

Cost drivers and normalization coefficients

2) Frequency of operation:

- Function of target frequency and inherent process speed (gate delay)
- The normalization cost drivers are non-linear.
  - Transistor counts are increased for higher frequency operation and decreased for lower.
  - The cost drivers typically range from .5 to 2.0 most complex designs utilizing N and N-1 processes.

3) Reuse:

- Function of the amount of FE and BE reuse.
  - Cost drivers range from: 1.0 for "no reuse" to .25 for reuse of FE data.
  - The smaller the cost driver the less the effort.

Cost drivers and normalization coefficients

4) Process utilization:

- Function of:
  - Transistor density per type of transistor
  - # metal layers
  - Pitch per metal layer,
  - How many devices have been manufactured in the process
- The normalization cost drivers are non-linear and depend
  - Transistor counts increase for higher cost drivers.
  - Cost drivers can typically range from .7 to 1.4 for nanometer processes.

Transistor Normalization Example

The transistor normalization equation is expressed as:

$Trans_{norm} = Trans_{Tot} * f(type) * f(freq) * f(reuse) * f(density) * ??? * ???$

Normalization is done on a block by block basis:

	Transistor Type			Transistor Normalization Cost Drivers				
Block	Random Logic Transistors	Structured Datapath Transistors	Memory Transistors	Circuit Type	Operation Freq	Reuse	Process Utilization	Total Transistors Normalized
BTC	43,127	89,560	256,000	1.2 0.98 0.6	1.3 1.3 1.1	1 0.25 0.25	0.96 0.94 1.4	**64,587** **26,813** **59,136**
ALU	5,490	45,898	0	1.2 0.98 0.6	1.3 1.3 1.1	1 0.25 0.25	0.96 0.94 1.4	**8,222** **13,741** **0**
Inst Decode	95,000	65,234	455,000	1.2 0.98 0.3	1.3 1.3 1.1	1 0.25 0.12	0.96 0.94 1.4	**142,272** **19,530** **25,225**

Productivity FOM

- There are number of environmental and methodological cost drivers which impact productivity. These include:
  - High level language modeling
  - Debugging environment
  - EDP methodology
  - Front-end and back-end design & verification methodology
  - Designer experience and capabilities
  - Management experience and capabilities
  - Legacy design issues
  - Schedule pressures
  - Team locality
  - Funding issues
- The cost driver normalizes the actual productivity number.

Productivity Driver Example

Design	RTL Verilog	RTL Development Environment	Verification Environment	DFT Environment	Synthesis Environment	SC Based Design	Structured Custom Design	Data Path Design (Automated)	Datapath Design (Custom)	Schedule	Team Locality	Team Experienced	Productivity Driver Multiplier
High Level Model Design													
Memory	1.25	1.10	1.10	1.10	1.10		1.40	1.10	1.10	0.90	1.00	1.20	3.3
Execution	1.25	1.10	1.10	1.10	1.10		1.10	1.10	1.10	0.90	1.00	1.30	2.8
Front End	1.25	1.10	1.10	1.10	1.10		1.10	1.10	1.10	0.90	1.00	1.10	2.4
Sequencer	1.00	0.90	0.90	0.90	0.90		1.10	1.10	1.10	0.90	1.00	0.70	0.6
System	1.25	1.25	1.25	1.10	1.10		1.40	1.10	1.10	0.90	1.00	1.30	4.7
Gate Level Model Design													
Memory		1.10	1.10	1.10	1.10		1.40	1.10	1.10	0.90	0.90	1.10	2.2
Execution		1.10	1.10	1.10	1.10		1.40	1.10	0.70	0.90	0.90	1.10	1.4
Front End		1.10	1.10	1.10	1.10		1.40	1.10	1.10	0.90	0.90	0.90	1.8
Sequencer		0.90	0.90	0.90	0.90		0.90	1.10	1.10	0.90	0.90	1.10	0.6
System		1.25	1.25	1.10	1.10		1.40	1.10	1.10	0.90	0.90	1.10	2.9
Schematic level Design													
Memory						1.20	1.10	1.10	1.10	0.90	0.90	1.10	1.4
Execution						1.20	1.10	1.10	1.10	0.90	0.90	1.10	1.4
Front End						1.20	1.10	1.10	1.10	0.90	0.90	0.50	0.6
Sequencer						1.20	1.10	1.10	1.10	0.90	0.90	1.10	1.4
System						1.20	1.10	1.10	1.10	0.90	0.90	1.10	1.4
Mask Design						1.10	1.10	1.20	0.95	0.90		1.10	1.4
Library Design						1.30				0.90		1.10	1.3
Verification	1.10	1.20	1.10	1.10	1.20					0.90		1.10	1.9
CHIP Intergration	1.25	1.25	1.25	1.10	1.10			1.10	1.10	0.90		1.50	3.9

Productivity Driver Multiplier (PDM)

- The PDM is used to modulate the average productivity numbers that the organization has exhibited over a number of projects.
  - It takes about 3 projects for a decent set of internal productivity numbers can be obtained
  - It is possible to obtain industry averages from 3rd party consultants who will obfuscate the data to protect the identity of the sources.
  - Average productivity numbers exist for number of activities:
    - HLM Design: LOC/day
    - Gate Level Design: Gates/day -> mapped to Transistors/day
    - Transistor Level Design: Transistors/day
    - Layout Design: Transistors/day
    - Verification: LOC/day

Intersection of Productivity and T-NORM

- Normalized Productivity is derived by taking the normalized transistor productivity and multiplying it by the PDM
  - $Productivity_{norm}$ = ($Trans_{norm}$/person-week) * PDM
- Production rate is derived by taking the normalized transistor count and dividing it by the duration of the project.
  - Production Rate = $Trans_{norm}$/ week
- Both numbers can be generated for the various skills on the project:
  - RTL LOC $Productivity_{norm}$ = 160 LOC_{norm} /person-week
  - Logic design $Productivity_{norm}$ = 1145 $Trans_{norm}$ /person-week
  - Layout design Productivity Rate = 187 $Trans_{norm}$/week

Schedule Estimation

- The measured productivity and productivity-rate from past projects are then used to estimate schedules on future projects.
- This is accomplished by estimating the various transistor counts, transistor types, etc. for each block and then applying the productivity numbers. The results will determine the the approximate resource requirements and schedule implications.
- Additional cost drivers can also be applied if necessary at this time. These additional drivers can include:
  - Complexity
  - Methodology
  - Team capabilities
  - Team locality

Schedule Estimation Example

- The following spread sheet shows how to calculate the number of person weeks needed for a future project:

Block	Transistor Type				Productivity Normalization				
	Random Logic Transistors	Structured Datapath Transistors	Memory Transistors	Total Transistors Normalized	Average CD Productivity Trans/PW	PDM	Normalized Productivity	Total Person Weeks
BTC	43,127	89,560		**64,587**	1745	1.2	2094.0	**31**
				26,813	3430	1.3	4459.0	**6**
			256,000	**59,136**	4500	1.1	4950.0	**12**
ALU	5,490	45,898		**8,222**	1745	0.9	1570.5	**5**
				13,741	3430	1.0	3430.0	**4**
			0	**0**	4500	1.0	4500.0	**0**
Inst Decode	95,000	65,234		**142,272**	1745	0.9	1483.3	**96**
				19,530	3430	0.7	2229.5	**9**
			455,000	**25,225**	4500	1.0	4500.0	**6**

Product Definition Example

- Less complex factors are needed during the product definition phase:
  - Functional Block complexity
  - Frequency/Timing push w.r.t. process
- Estimates for new micro architectural features are based on data from comparable historical features weighted by cost drivers
  - Useful for evolutionary architectures
  - Must extrapolate new cost drivers for new architectures

Product Definition Example

Estimates for new micro architectural features are based on data from comparable historical features weighted by cost drivers.

Example:

New part "C" is designed with a 32 choose 8 distributed speculative scheduler. Historical data indicates that Part "T" has a similar, but simpler 16 choose 4 scheduler that had a normalized transistor count of 800K.

Part "C"s scheduler is different, and requires a new estimation.

Product Definition Example (cont)

Part "C"s scheduler is wider and more complex. Further, part "C" is higher frequency, utilizing a new process and more challenging circuit implementation techniques.

Historical comparisons can be used to evaluate these differences and estimate a complexity factor and frequency factor for the new design.

In this example Part "C"s estimate would have a complexity factor of 1.5 and a frequency factor of 1.4 and thus a Normalized TC of :

800K * 1.5 * 1.4 = 1.680M

Product Definition Example - Aftermath

Part "C"s scheduler requires over twice the normalized transistors to build.

The design team is now forced to consider:

- Is the performance benefit worth the implementation cost?
- Are there simpler methods that still yield acceptable performance?
- Is there a better performance/implementation cost point?

Enabling rigorous complexity analysis and control in the EDP stages of product development will incentivize architects to innovate less complex solutions to performance problems.

Other Cost Driver Examples

Cortex A9 Core - 45nm

Metrics	Baseline	Freq	ASIC Standard Cell Library	Rvt	Pulse-flop	Domino-flop	NDL	High Perf Memory	RTL opt	Modified Result	
Frequency (MHz)	600	1.00	0.97	1.00	1.05	1.05	1.00	1.02	1.14	747.94	MHz
Energy (mW/MHz)	0.5	1.00	0.95	1.00	1.05	1.05	1.00	1.00	1.00	0.52	mW/MHz
Area (mm2)	3.00	1.10	1.00	1.00	1.02	1.03	1.00	1.02	1.00	3.50	mm2

Cortex A9 Core 32nm

Metrics	45nm Baseline	32nm Process Improvement	Freq	HP Library	LVT	Pulse-flop	Domino-flop	NDL	Custom	RTL opt	ARM RTL reuse	Multi PVT sign-off corners	Modified Result	
Frequency (MHz)	600.00	1.21	1.00	1.10	1.10	1.05	1.05	1.17	1.00	1.10	1.00	1.00	1246.46	MHz
Energy (mW/MHz)	0.50	0.61	1.00	1.05	1.00	1.05	1.05	1.10	1.00	1.00	1.00	1.00	0.39	mW/MHz
Schedule (months)	6.00	1.00	1.17	1.00	1.00	1.10	1.10	1.10	1.00	1.00	0.83	1.17	9.07	Months
Area (mm2)	3.00	0.45	1.21	1.16	1.00	1.06	1.06	1.15	1.00	1.00	1.00	1.00	2.45	mm2

Cortex A9 Core 28 nm

Metrics	Baseline	28nm Process	Freq	HP Library	<15% LVT	Pulse-flop	Domino-flop	NDL	Custom	RTL opt	Multi PVT sign-off corners	Modified Result	
Frequency (MHz)	600.00	1.25	1.00	1.08	1.13	1.05	1.05	1.15	1.00	1.10	1.00	1270.46	MHz
Energy (mW/MHz)	0.50	0.52	1.00	1.05	1.10	1.05	1.05	1.10	1.00	1.00	1.00	0.36	mW/MHz
Area (mm2)	3.00	0.27	1.21	1.16	1.00	1.06	1.06	1.15	1.00	1.00	1.00	1.47	mm2

Agenda

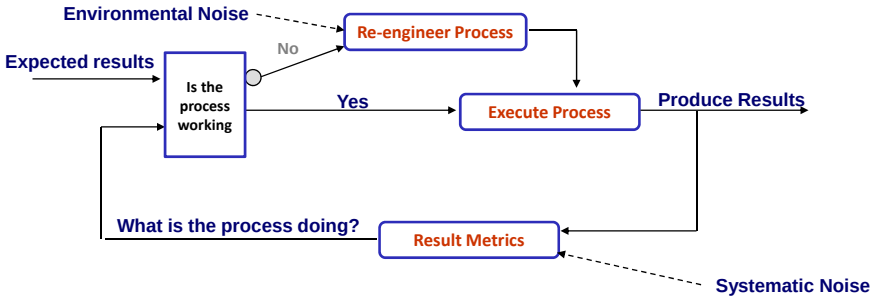
- Design metrics
- Complexity Models
- SW Metrics
- HW Metrics
- Process Flow Metrics

Process vs. Product Metrics

- Process Metrics-
  - Insights of process paradigm, software engineering tasks, work product, or milestones.
  - Lead to long term process improvement.
- Product Metrics-
  - Assesses the state of the project
  - Track potential risks
  - Uncover problem areas
  - Adjust workflow or tasks
  - Evaluate teams ability to control quality

Process Metrics

- All processes must be monitored.
- Processes must be closed loop.



- Processes can be over damped or under damped.
  - Need to validate what the indicators are telling you.
  - How do you adjust an out of control process?