On the Energy Consumption of Design Patterns

Christian Bunse – Fachhochschule Stralsund
Sebastian Stiemer – Fachhochschule Stralsund

EASED@BUIS 2013

Oldenburg, April 2013

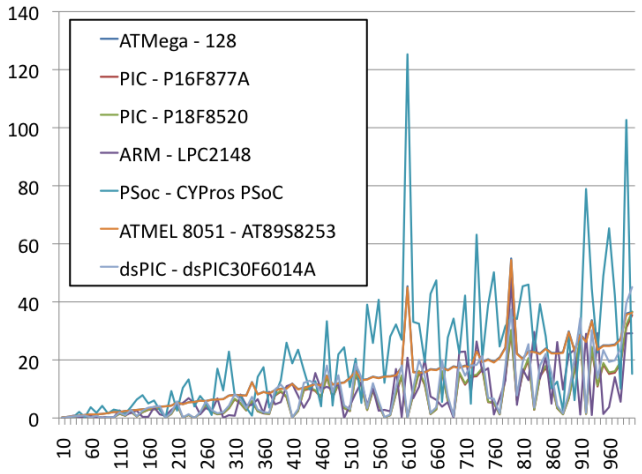
fachhochschule stralsund
university of applied sciences
fachbereich elektrotechnik + informatik
school of electrical engineering + computer science

Motivation

- Standard personal computer (PC)
  - Typical energy consumption: ~150 Watt → 125 kWh per year
  - In Europe 62% of the inhabitants and 97% of companies make use of one or more computing systems
    - Europe consumes several TWh for computing
    - In addition there are more than 400 million mobile phones in the EU, …
  - Estimation: in 2020 ~35% of the global energy consumption are needed for computing
- Other factors
  - Cloud computing, social communities, online gaming, etc.
  - A Google search causes a CO_2 emission of 2-10g [Wissner-Gross]
- Research often has a focus on hardware
  - Green-IT, Low-Power Design, “Dynamic Voltage Scaling”, …
- However, software has an impact, too?

Software and Energy

- Case studies show:
  - [Huels2002] Replacing double by float leads to a 34% lower energy consumption and to 35% less cycles
  - [HoeBu2010] Algorithms do have a unique energy signature:



  - [HoeBu2010] Resource-substitution significantly reduces energy needs
  - [HoeBu2011] Signature-based optimisation further reduces energy demands

From Algorithms to higher Abstraction Levels

- Patterns
  - name, abstract and identify key aspects of solutions to recurring problems
  - Are often recommended as best practice
    - Reusability
    - Flexibility
    - …
  - The impact of pattern onto quality factors such as performance, security or energy consumption is often unknown

- Evaluate the impact of Patterns onto energy consumption

Related Studies

- Litke et al
  - „... analyze six design patterns and explore the effect of them on energy consumption and performance."
  - Microprocessor Platform & C++
  - Results
    - "We have observed that except for one example where the energy consumption of the pattern solution increased significantly (and thus making the application of the pattern questionable from a power point of view), the use of design patterns does not necessarily impose a significant penalty on power consumption."
    - Reason: Increased memory and processor usage

Related Studies

- Sahin et al.:
  - „... compare the power profiles of software using design patterns against software not using design patterns ...”
  - Analysis tool for UML based profiles
  - Results
    - applying design patterns can both increase and decrease the amount of energy used by an application;
    - design patterns within a category do not impact energy usage in similar ways;
    - it is unlikely that impacts on energy usage can be precisely estimated by only considering design level artifacts.

Intermediate Conclusion

- A design pattern is not a finished design that can be transformed directly into source or machine code.
  - Template for problem solving
  - Must be individually implemented
  - ➢ Can we extrapolate general energy needs?
- So far, isolated results for a dedicated context
- Replication under varying conditions needed
  - Mobile Systems
  - Java
  - …
- Goal: Meta-analysis.

Our Approach

- Goal: Replicate results of prior studies on a mobile platform
  - Android
  - Java
- Select: Choose patterns
  - Family, popularity, etc.
- Approach: Run, Measure, and Compare
  - Isolate effects
  - Avoid interference
- Measurement:
  - Software-based (PowerTutor)

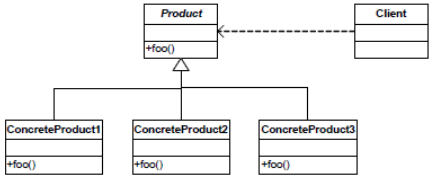
Patterns Examined

Experiment	Pattern	Category
1	Facade	Structure
2	Abstract Factory	Creator
3	Observer	Behavior
4	Decorator	Structure
5	Protoype	Creator
6	Templatemethod	Behavior

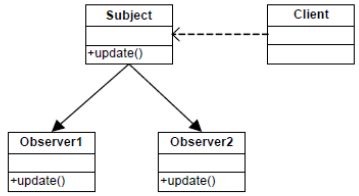
- Based on Gamma et al
- Most popular patterns [Developer.com]
-

Setup & Systems

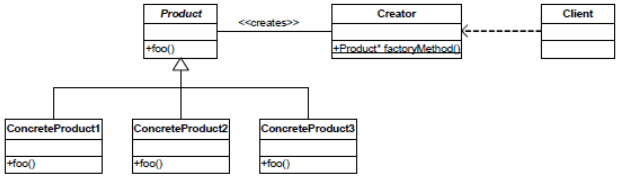
- Experiment Framework
- Two versions for each pattern (following Litke et al and other)

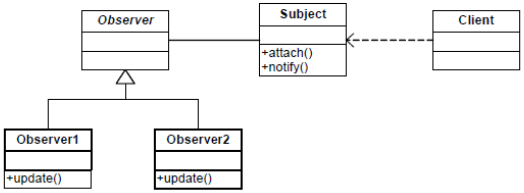


Non-Pattern Solution

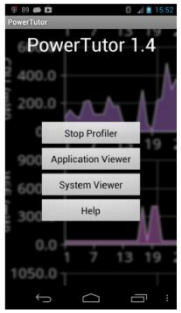


Non-Pattern Solution





“Doing the Measurements”



- Prerequisites
  - Profiler-Service (PowerTutor)
  - Unused components (GPS, Display) are switched off
- Execution
  - Experimental subject is started by using the ECLIPSE/ADB
  - Updates are noted in Logcat
  - End of run is signalled
  - Powertutor provides energy consumption information
- Outlier and data validity
  - Measurements were replicated (>100) and results were normalised
  - Performed at varying times (temperature)
- Threats to validity
  - Just a single device (Galaxy Nexus)
  - No threading, multicores, etc.
  - Java was used as main programming language (impact of VM?)

Results

				Difference (%)	
Pattern	System	Overall Time	Energy (J)	Time	Energy
Facade	"Clean"	15,40	395,60	1,80	2,50
	Pattern	15,70	405,60		
Abstract Factory	"Clean"	13,50	342,10	14,20	15,90
	Pattern	15,40	396,60		
Observer	"Clean"	15,10	373,70	0,90	0,10
	Pattern	15,20	373,90		
Decorator	"Clean"	15,20	374,00	132,40	133,60
	Pattern	35,40	873,80		
Prototype	"Clean"	11,20	271,80	33,00	33,20
	Pattern	14,90	362,00		
Template Method	"Clean"	15,00	366,40	0,30	0,10
	Pattern	15,10	366,70		

Discussion

Overall measurement results are not surprising ☹, but confirm

- Load and store instructions require more cycles per instruction and in addition have a larger energy cost than arithmetic or branch instructions:
  - memory consumption is of equal importance as processor cycles
- performance optimization will improve energy consumption but effects are smaller than predicted
- the OS has an impact that needs to be cleanly addressed

Summary & Conclusion

- In general our experiments have confirmed prior studies in that
  - applying design patterns can both increase and decrease the amount of energy used by an application;
  - design patterns within a category do not impact energy usage in similar ways; and
  - it is unlikely that impacts on energy usage can be precisely estimated by only considering design level artifacts.
- Memory access seems to be the key cost driver, but what about the
  - Garbage collector?
  - App Lifecycles
  - Multicore, Threads, External Storage,

Future Work

- Replicate using more devices
- Support multi-core systems
- Provide measurement opportunities for other components
- Vision: Integrated measurement tool for mobile systems