Transparency

CS580 Advanced Software Engineering
http://cs580.yusun.io
October 15, 2014

Yu Sun, Ph.D.
http://yusun.io
yusun@csupomona.edu

CAL POLY POMONA

The Paper

- "Use of the Concept of Transparency in the Design of Hierarchically Structured Systems"
- David Parnas and Daniel Siewiorek
- Communications of the ACM, 1975

Top Down Design (*aka* Outside In)

- Begin with precise specification
- Derive internal structure from the specification

- Difficult or infeasible to obtain full specification
- The derivation of the design is often infeasible
- Can result in software that is unnecessarily inflexible (difficult to derive architecture for a product line)

- For these reasons, pure Top Down has problems

Bottom Up Design

- Assume a well defined lower level
- Consider the design of the next highest level

- Create the system "Inside Out" from a set of lower level components (i.e. "start at the bottom")
- Work upwards, solving entire project
- Reuse components from other projects
- More practical to implement internal structures first, creating separate modules and joining them together
- Bottom Up is more flexible. Hard to design "general purpose" system / library using top-down

Bottom Up Design

- Base Machine
  - The lower level of a hierarchy, maybe hardware or an intermediate software level
- Virtual Machine
  - A level above the base machine, it hides the complexity of the base machine to make interaction with the system easier

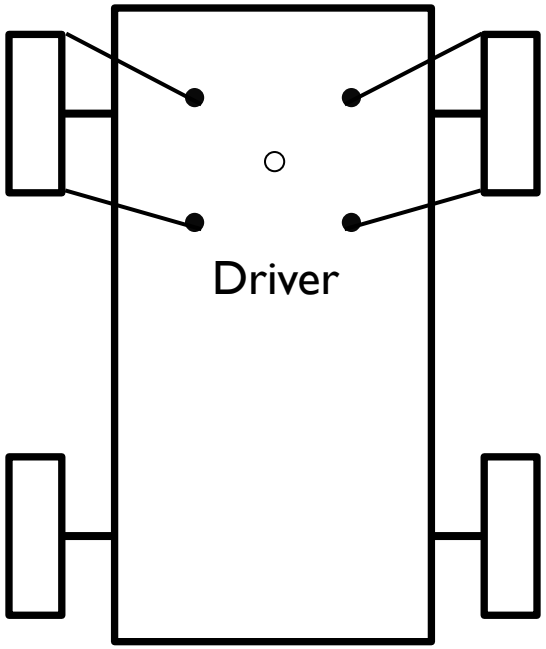
Transparency in Bottom Up Design

- Transparency
  - Describes the implementation completeness of the virtual machine with respect to the base machine's functionality
- Complete transparency
  - The virtual machine has ALL of the functionality of the base machine
- Loss of transparency
  - A lack of functionality with respect to the base machine exists in the virtual machine
  - There is some sequence that can be specified in the base machine that can not be expressed in the virtual machine
  - "Often one of the goals of a design"

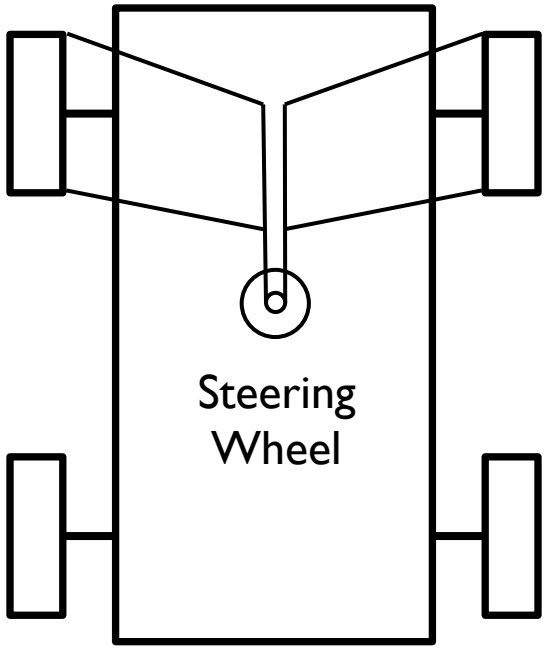
Examples

- Java virtual machine
  - Underlying hardware may have more capabilities than the JVM allows access to
- Volume on stereo system
  - Speakers may be able to produce higher volume than stereo allows; protect from speaker blowout
- Hot water heater and faucet
  - Hot water heater can deliver scolding hot water; faucets have a beneficial loss of transparency to protect from burns

Driving with Strings and Steering Wheel

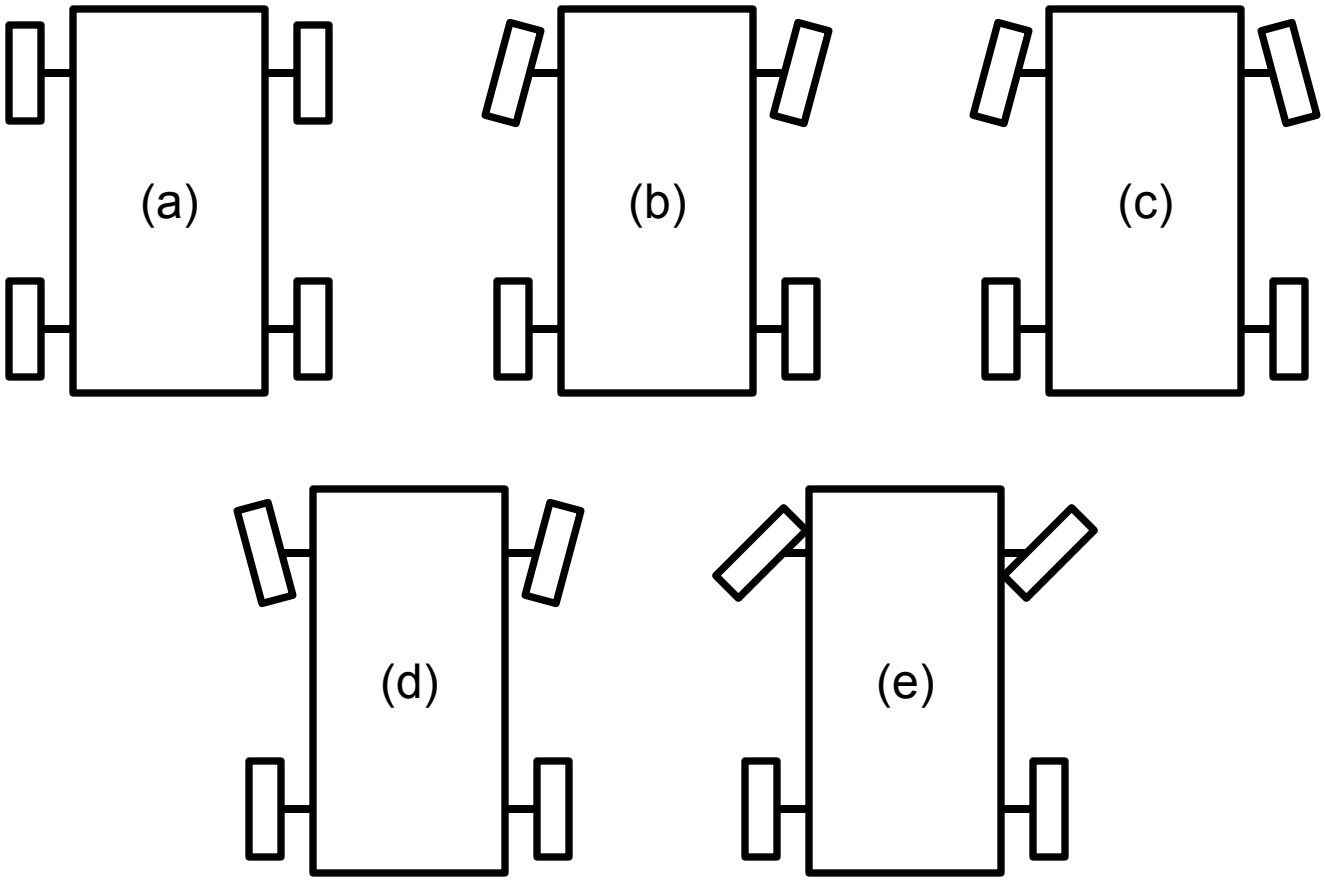


Base machine



New virtual machine

Example Positions



What figures suggest a loss of transparency?
In this case, is the loss of transparency ok?

Validity Based on Transparency

- A purpose of introducing abstractions is to prevent undesirable states occurences
- Examine the loss of transparency in each level
  - Nothing desirable is lost
  - Assured that upper levels still have the desired capabilities

Other Examples of Transparency

- Hardware
- Search Engine
- Layered debugging for DSLs

Example 1 – Graphics Card Transparency

Hierarchical Level	Description
3	Application – Game, CAD
2	API – DirectX,OpenGL
1	Driver
0	Graphics Card – silicon

Graphics Card Example

- Positive results of transparency
  - Much easier to program with API than directly with driver
  - Using an API lets an application run on different hardware
- Negative results of transparency
  - Depending on implementation, an application might not run as fast on a particular piece of hardware
    - i.e., it won't fully utilize certain hardware features

Nvidia's Unified Driver Architecture

- A way of dealing with a changing base machine

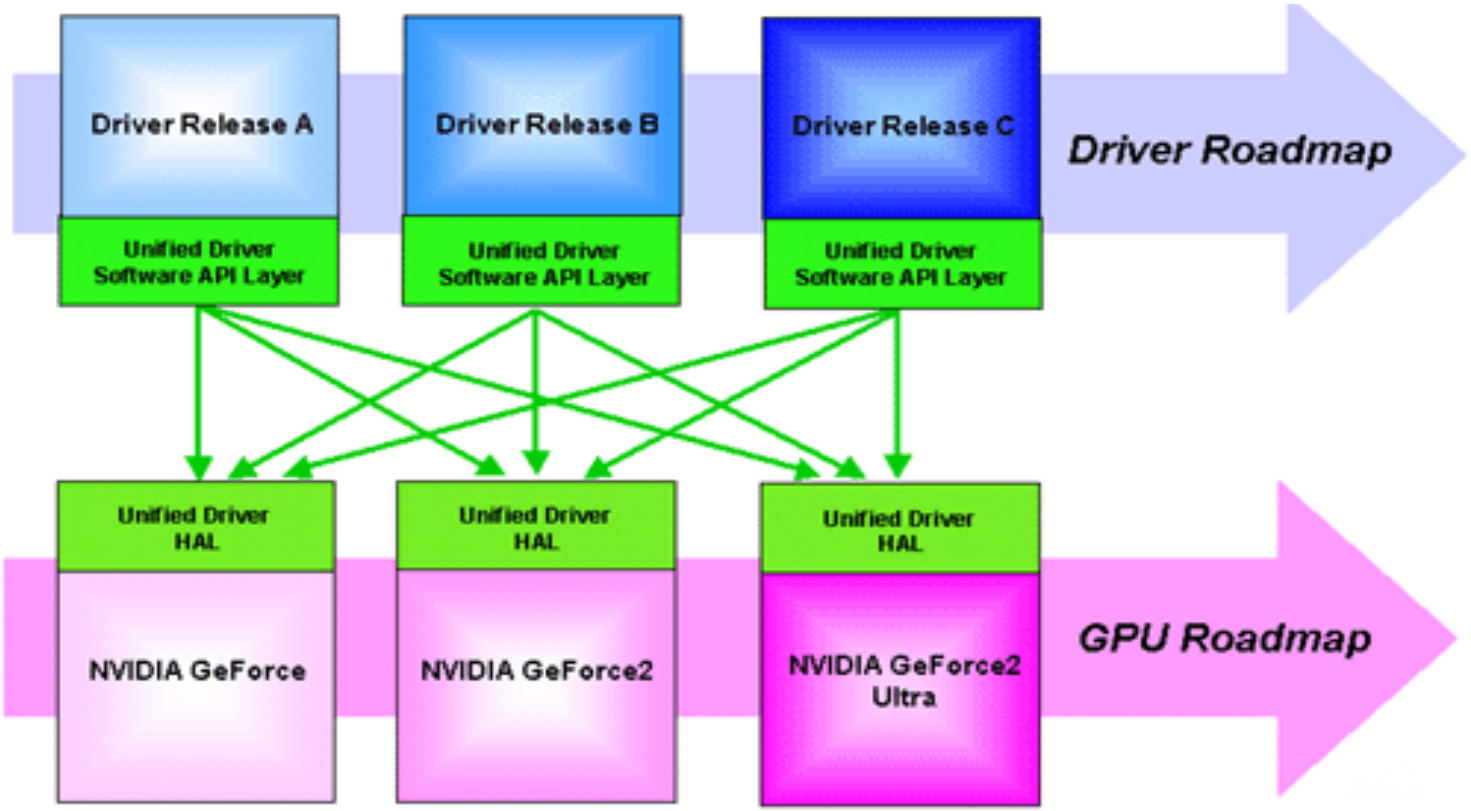


Image courtesy of AnandTech

Anand Lal Shimpi, "NVIDIA's Detonator3 Drivers - Teaching an 'old' dog new tricks", August 14, 2000

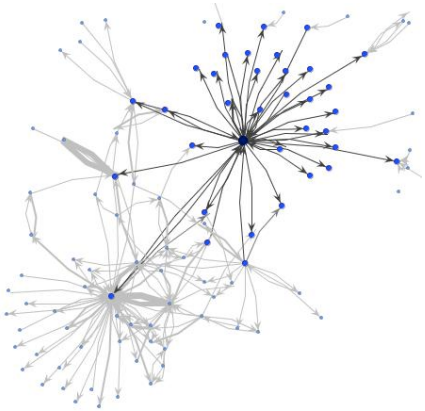
Nvidia's Unified Driver Architecture

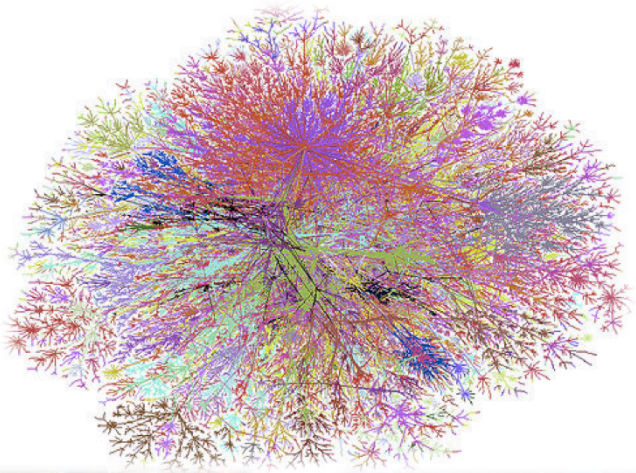
- All drivers communicate with a hardware abstraction layer (HAL) that resides on silicon "temporary degree of transparency"
  - Basic functionality exists with older drivers
  - Higher performance arrives with newer drivers

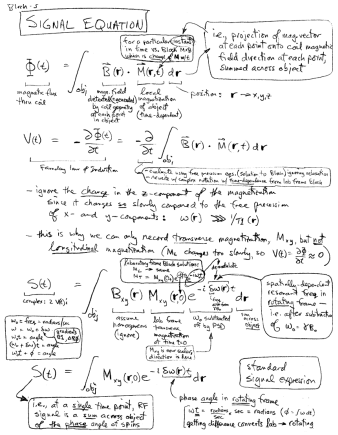
Example 2 - Search Engine Example

- Simple user interface which interacts with a complex database
- User can only run simple queries
- User cannot modify data through the interface

What's behind Google Search?



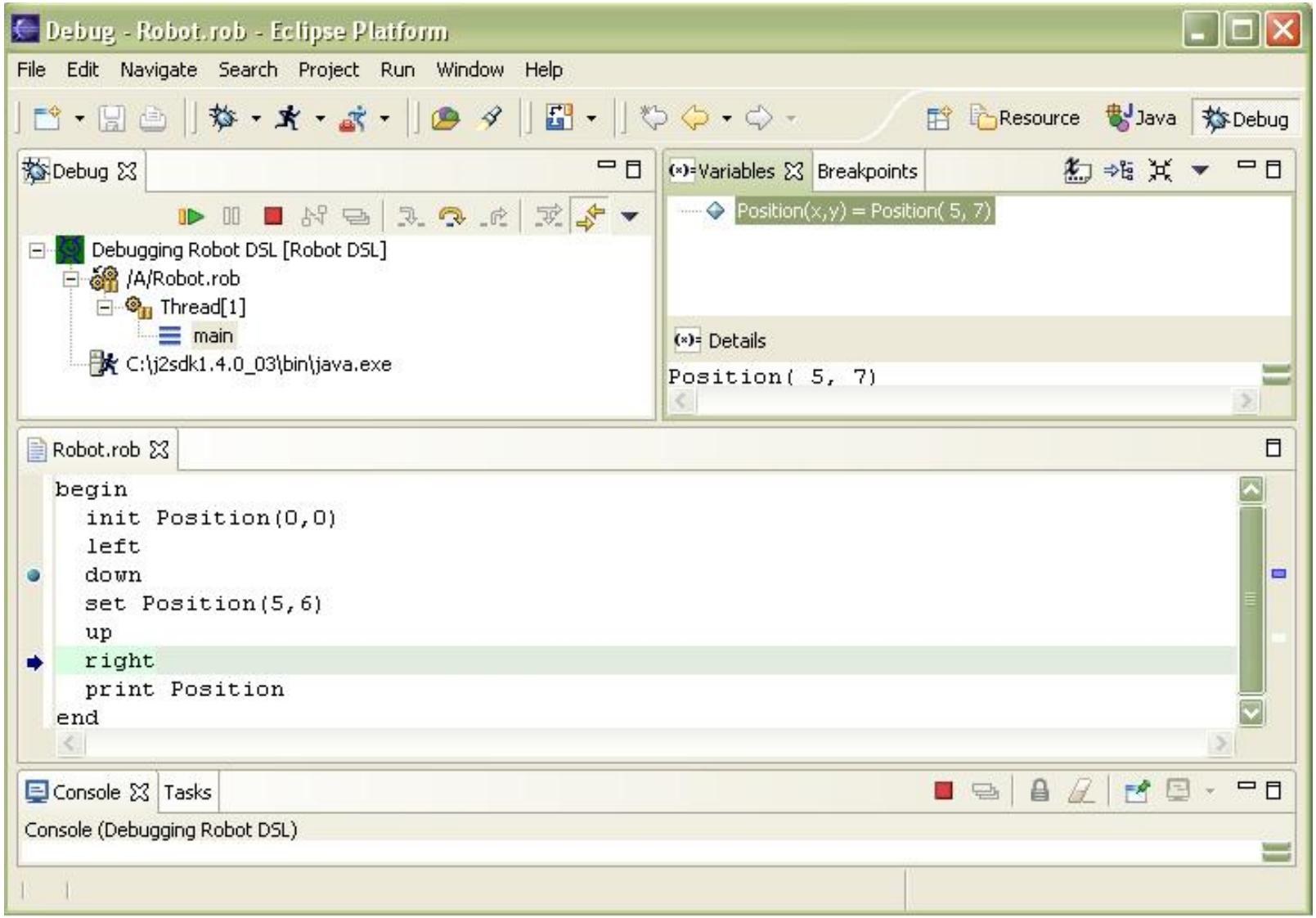




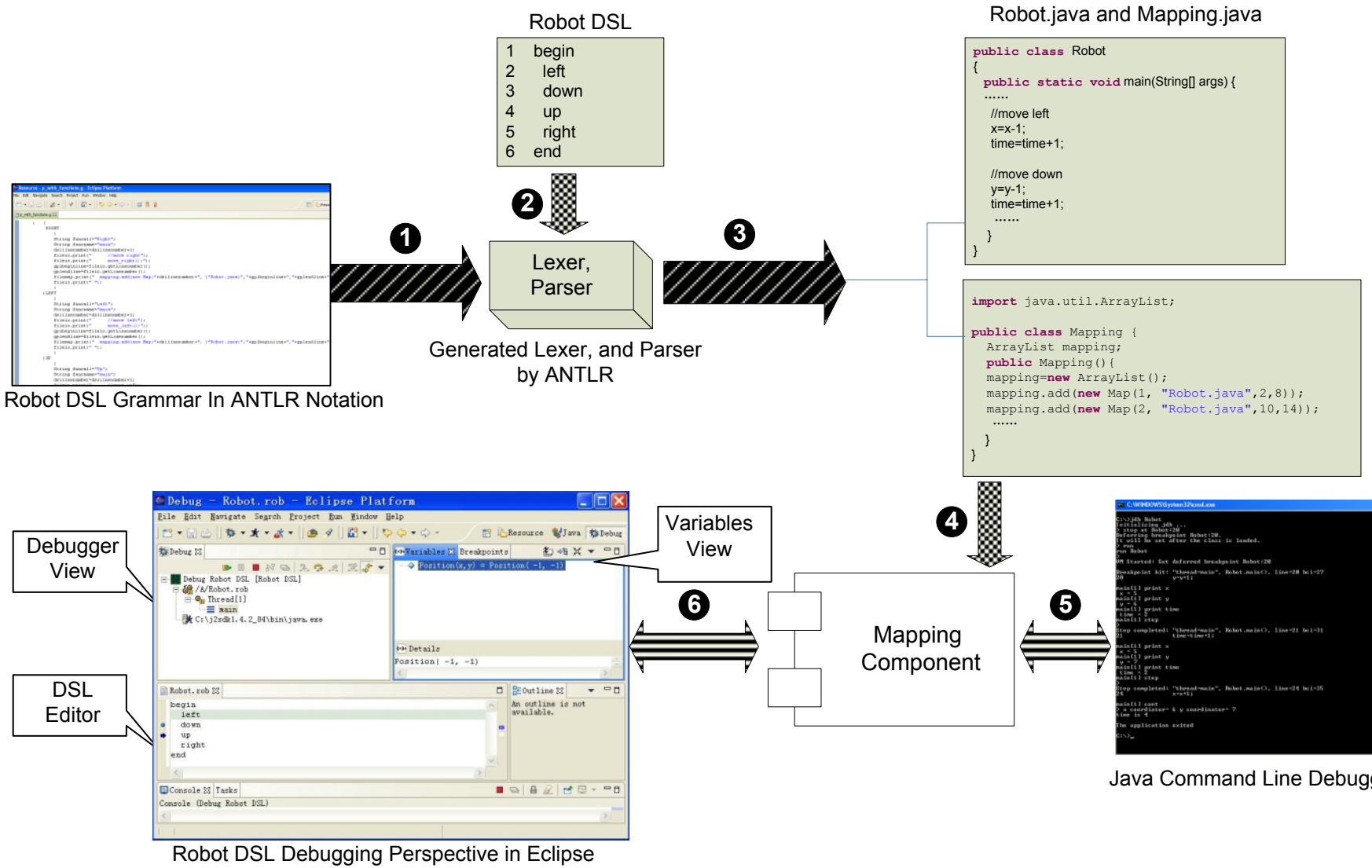
Search Engine Example

- Large loss of transparency yields mainly positive effects
- Maintains data integrity
- Loss of transparency does not hinder usability from the user's perspective
- Few significant negative effects can be identified

Example 3 - DSL Debugger in Eclipse



DSL Debugger Transparency



Back to Parnas: Suggestive and Misleading Transparency

- Suggestive
  - Sometimes the virtual machine should give "suggestions" to the base machine
- Misleading
  - Sometimes the virtual machine implementation is inefficient because of a lack of information at a lower level

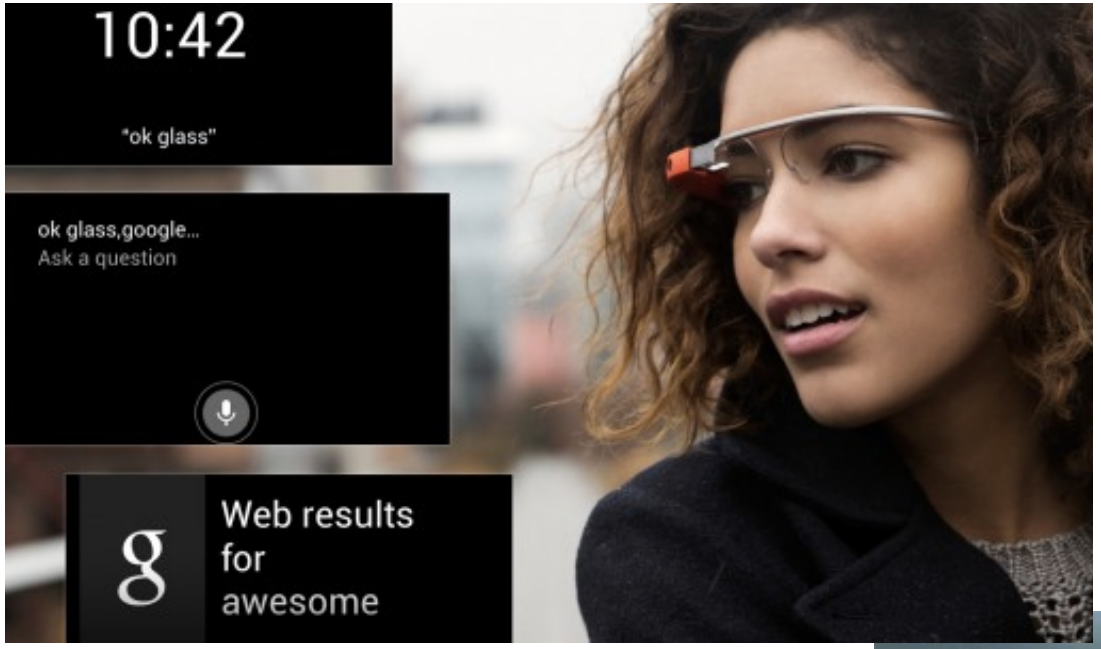
Conclusion

- "… a fundamental 'tradeoff' which exists between transparency and flexibility of a design"
- Determining the right level of transparency for the higher level structure is crucial in order to obtain its goals

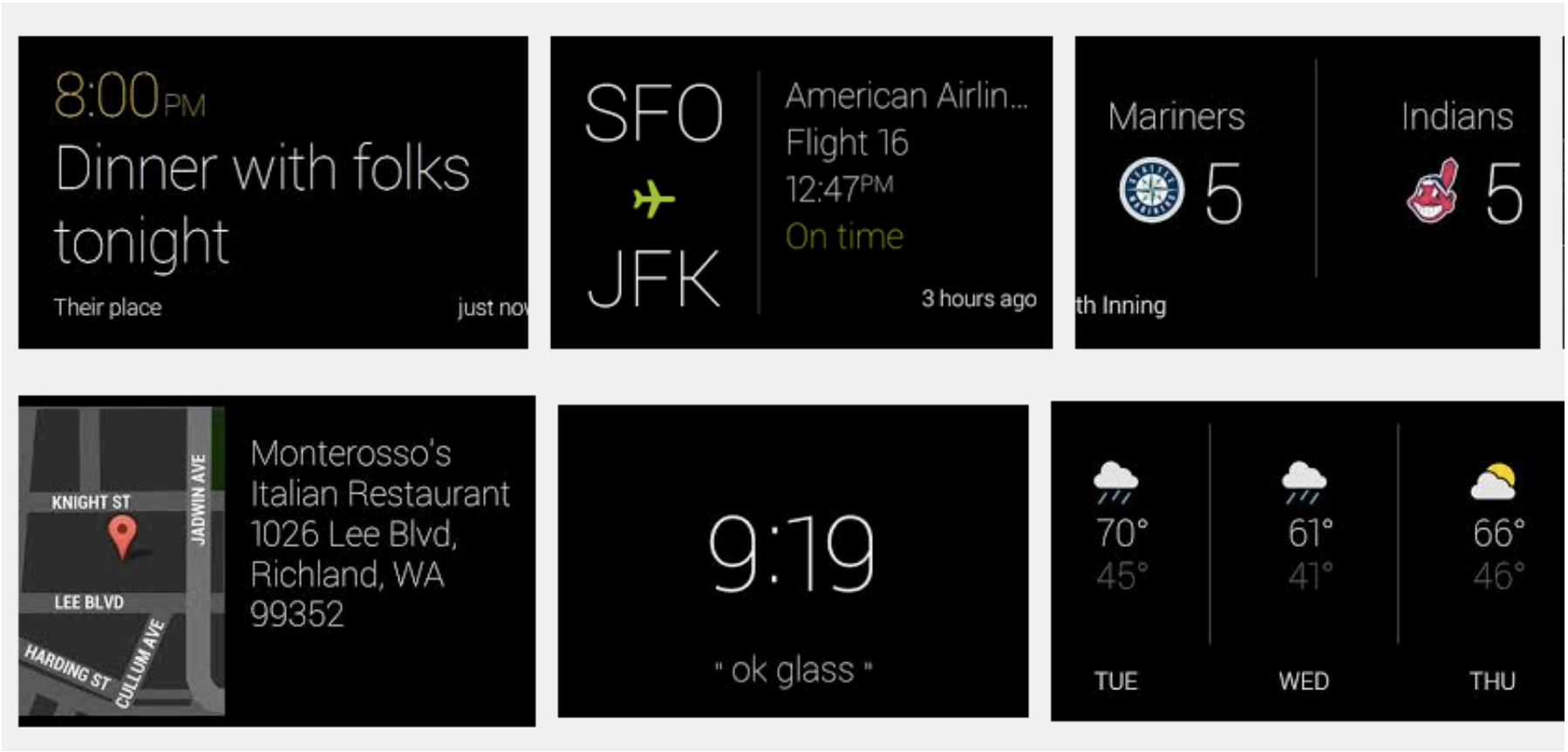
Outside in + Bottom up + Agile

- Know the high level specs
- Evaluate the feasibility
- Iterative development and refinement

AR in Google Glass



Google Glass Card UI



Outside in + Bottom up + Agile

- Positive results of transparency
  - Easy APIs to build UI "cards"
  - Easy integration with other services (Google Hangout, Camera, Authentication)
  - Fast and easy communication with cloud servers
- Negative results of transparency
  - Everything is done in cloud
  - Glass only handles web-based rendering and interaction
  - No heavy computations can be done in Glass
  - Must follow the base UI and workflow