



# Arquitecturas y Organizaciones

Idilios y desencuentros

Hablo con mis amigos  
para saber lo que pienso

Miguel de Unamuno

# Algunas ideas

- No todas las arquitecturas son valiosas en todos los contextos.
- No todas las organizaciones pueden usar/producir todas las arquitecturas
- Hay un nivel máximo de complejidad tolerable/eficiente para cada organización (Juan Gabardini)
- Podemos usar criterios de usabilidad para entender si las arquitecturas, como las cosas, son razonablemente fáciles de usar.

# Grandes ideas

- La estructura del sistema se corresponde con la estructura de la organización (Conway)
  - Organization follows location
- y
- Code Ownership (Coplien y Harrison)

# Otras grandes ideas

- Un módulo es arquitectónicamente significativo si tiene un bajo número de camión

(Bass, Clements, Garlan et al)

- Módulo como “responsibility assignment rather than a subprogram”

(Parnas)

# Otra más

“Organization-architecture conformance is largely a matter of whether the implied communication pathways are consistent with the software dependency graph”

Jim Herbsleb,  
en palabras de David Garlan

# Estructuras de un sistema

- Tipos de estructuras

- Módulos (Código)

- Ejecución

- Ubicación (Software vs. el mundo)

(Bass, Clements, Garlan et al)

# ¿y la colaboración?

La modularidad permite proteger el trabajo de equipos distintos pero limita las oportunidades de colaboración entre equipos

(Austin y Devin)

# El tamaño de nuestras abstracciones crece constantemente

Mary Shaw,  
Three patterns that help  
explain the development of  
software engineering

# Un ejemplo por favor

- Reuso
  - Líneas de productos
    - SOA
- PHP vs.
- Cluster
- PaaS
- Monitoreo vs. Diseño

# Usabilidad de las arquitecturas

¿Podemos aplicar principios y heurísticas de usabilidad para evaluar si una arquitectura le conviene a una organización?

# Menos es más

- Simplicidad (KISS)
- +Indirección/+Complejidad
- Si queremos menos complejidad, tenemos que tener menos requerimientos (Poppendiecks)
- ¿Realmente necesitamos una arquitectura de 5 capas?
- ¿Cuánto hay que aprender antes?
- ¿Estamos dispuestos a aceptar que nos queda grande?

# Correspondencia Natural/ Affordance

- ¿La arquitectura permite cambiar lo que varía y no cambiar lo que no varía?
  - “Where do architectures come from? They spring from the minds of the architects” (Clements)
- ¿La arquitectura que usaron con éxito les parece apropiada para todo?

# Modelos Mentales Consistentes

- ¿Hay una metáfora del sistema?
- ¿Se mantiene la integridad conceptual del sistema?
- ¿La estructura lógica se corresponde con la de módulos? (Meyer, Unidades Modulares Lingüísticas)
- Si es un enlatado o software libre ¿Cambiamos nuestro modelo mental?
- ¿Qué departamentos involucra?

# Tolerar errores

- ¿Cuál es el impacto de hacer modificaciones sin comprender el sistema?
- Principio de Protección (Meyer) ¿Hasta donde llega un error?
- ¿Cambiar la configuración requiere reiniciar la aplicación?
- ¿Qué impacto tienen las inconsistencias?

# Funciones restrictivas

“C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off.”

Bjarne Stroustrup

- Frameworks
- Generación de código
- Puntos de extensión
- Código aislado (NO TOCAR)

# Devolución (Feedback)

- Monitoreo y visibilidad
- Métricas de atributos de calidad (¿Es fácil de probar?)
- De cerca ¿Son evidentes las ventajas y desventajas? (Sin mirar el folleto)
- Si la usamos inapropiadamente ¿Tiene algo que decir al respecto?

# ¿Entonces?

- ¿No todas las arquitecturas son valiosas en todos los contextos?
- ¿No todas las organizaciones pueden usar/producir todas las arquitecturas?
- ¿Hay un nivel máximo de complejidad tolerable/eficiente para cada organización? (Juan Gabardini)
- ¿Podemos usar criterios de usabilidad para entender si las arquitecturas, como las cosas, son razonablemente fáciles de usar?

¿Conclusiones?  
¿Devoluciones?

# Bibliografía y Referencias

- Austin, Robert, Devin, Lee, ***Artful making***, 1993
- Bass L., Clements P., Kazman R.; **Software Architecture in Practice - 2nd Edition**, Reading MA: Addison-Wesley, 2003.
- Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; & Stafford, J. **Documenting Software Architectures: Views and Beyond**. Boston, MA: Addison-Wesley, 2002.
- Coplien, James, Harrison, Neil B., **Organizational Patterns of Agile Software Development**, Prentice Hall, 2005.
- Nielsen, Jakob, **Usability Engineering**, Morgan Kaufmann, 1993.
- Norman, Donald, **The Design of Everyday things**, MIT Press, 1998.

# Bibliografía y Referencias

- Shaw, Mary, **Three Patterns that help explain the development of software engineering**, Position paper for Dagstuhl Workshop on Software Architecture, August 1996.
- Stroustrup, Bjarne, **“Did I really say that”**, [http://www2.research.att.com/~bs/bs\\_faq.html#really-say-that](http://www2.research.att.com/~bs/bs_faq.html#really-say-that), accedido 27/10/2011.