The System Shell as a Construct for Mitigating the Impact of Changing Contexts by Creating Opportunities for Value Robustness

1st Annual IEEE Systems Conference
Honolulu, HI, April 9-12, 2007

Presented by Adam Ross, Ph.D.
April 11, 2007
Achieving System Success

• Goal of design is to create value (profits, usefulness, voice of the customer, etc…)
• Requirements capture a mapping of needs to specifications to guide design
Deploying a “Valuable” System…

Contexts change…
Achieving System Success (cont.)

- Goal of design is to create value (profits, usefulness, voice of the customer, etc…)
- People change their minds; the world changes…
- To continue to deliver value, systems must cope with context change…
What is Context?

• Context includes forces exogenous to system
  – Stakeholder expectations
  – Operating environment
  – Policy constraints
  – Available technologies
  – Competitive market
  – Etc…

• System success depends on system performance within a given context

• In order to ensure success, designers must consider context beyond traditional “operating environment” (classical robust design)
What is System Success?

Success is defined across multiple perspectives and multiple time periods.

System success, $\Psi$, across N decision makers at time t

$$\Psi(t) = \sum_{i=1}^{N} [X_{DMI}(t) + \varepsilon_{C}^{X_{DMI}}(t)] \geq Y_{DMI}(t) + \varepsilon_{C}^{Y_{DMI}}(t)$$

$$0 \leq \Psi(t) \leq N$$

- $X_{DMI}(t)$: Decision maker i system “experience” at time t
- $Y_{DMI}(t)$: Decision maker i system “expectation” at time t
- $\varepsilon_{C}^{X_{DMI}}(t)$: Context effect on decision maker i “experience” at time t
- $\varepsilon_{C}^{Y_{DMI}}(t)$: Context effect on decision maker i “expectation” at time t

System Success: Net “experience” must meet or exceed net “expectations”
Dynamic System Context: Value Lenses

Static View
Structural Metrics

Dynamic View
Operational Metrics

Constraints

System Boundary

"Decisional"
Value

"Expectations"

"Experienced"
Value

"Experience"

"Decision", "Experienced", and "Remembered" Utility from (Kahneman and Tversky 2000)

"Discussion of "structural" versus "operational" metrics in (Giachetti et al. 2003)
Construct: Black Box System
Black Box System

The box has attributes (function and/or form) that are value-perceived.

“Experienced” Value

“Decisional” Value
Value Robust Black Box

Box is perceived to deliver the same value

The box is robust if it can maintain its value delivery as external “context” changes
Robustness Defined

A box can be quantified in terms of robust in $X_i$ to “Input” change

(i.e., can $X_i$ remain “constant” over range of “Input”?)

Level of attribute performance is function of inputs (and constraints including environment), so robustness is an insensitivity to the inputs (and constraints)
System Shell Illustrated

Inner Shell (The Shelter)
Change Context as seen by Box

Outer Shell (The Mask)
Change Box as seen by Context

The shell **protects** the box from changes in its context

The shell **filters** the appearance of the box to its context
Examples: Shelter

Context: Noisy environment

Solution: Earplugs

Context: Hostile external network

Solution: Firewall

Shelters “protect” system from context more cheaply than modification of system itself
System Mask as a Filter

User desires “A”
User sees “A”
Success!

User desires “AB”
User sees “AB”
Success!

User desires “B”
User sees “B”
Success!

System Mask customizes “experience” to meet expectations
Examples: Mask

Cellular Phone “Appearance”

GPS System “Appearance”

Solution: Faceplates

Solution: Various interpretive receivers


Photo courtesy U.S. Department of Defense

Photo courtesy Garmin

Masks filter “appearance” of system to context more cheaply than modification of system itself
Examples: Shell

Changing Context: International Electrical Power Sockets

Plug in device must “look” different to each electric socket

Device must be “protected” from varying voltage at each electric socket

Source: http://www.kropla.com/electric2.htm
Examples: Software Shell

Changing Context: Software Models to be Linked for Analysis

Plug-in software models must “look” the same to the Analyst

Plug-in software models must be “protected” from varying input/output formats

Mask: standardize code appearance to ModelCenter

Shelter: provide proper code input translation from ModelCenter

Result: “black box” software code; model plug-in capability

Example: Software/SoS Shell

<table>
<thead>
<tr>
<th>Component set</th>
<th>Shelter custom translation code</th>
<th>Mask standardized format</th>
<th>Appearance to SoS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(virtually limitless)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Component set** (virtually limitless)
- **Shelter** custom translation code
- **Mask** standardized format
- **Appearance to SoS**

Result: “system of system” design with standard component appearance
Flow of Value Robustness

Robustness strategies can be pursued at various locations in value flow from context to value perception.
Discussion

• When should a shell be used?

\[
\text{Benefit}_{\text{old\_sys+shell}} - \text{Cost}_{\text{shell}} > \text{Benefit}_{\text{new\_sys}} - \text{Cost}_{\text{new\_sys}}
\]

i.e., Net Benefit of Shell > Net Benefit of New System

• Implications for design
  – Extend operating ranges (augmented robustness)
  – Customized user experiences (multi-stakeholder satisfaction)
  – Distribution of costs for “shell”

• System of System design

• Separating changeable system parts
Conclusion

• System Shell construct decouples system from context through “protection” and “filtering”
• System Shell can be part of system or layered on top
• May be a cost-effective and/or timely solution to multiple perspective, multiple time period context

Using a System Shell is an effective technique for achieving Value Robustness
Thank you for your attention!

Any questions?

For further details on topic please see: