ISR Institute for Software Research
UNIVERSITY OF CALIFORNIA, IRVINE

Knowledge-based Architectural Adaptation Management (Northrop Grumman)

John Georgas

Institute for Software Research

University of California, Irvine

June 28, 2005

http://www.isr.uci.edu/

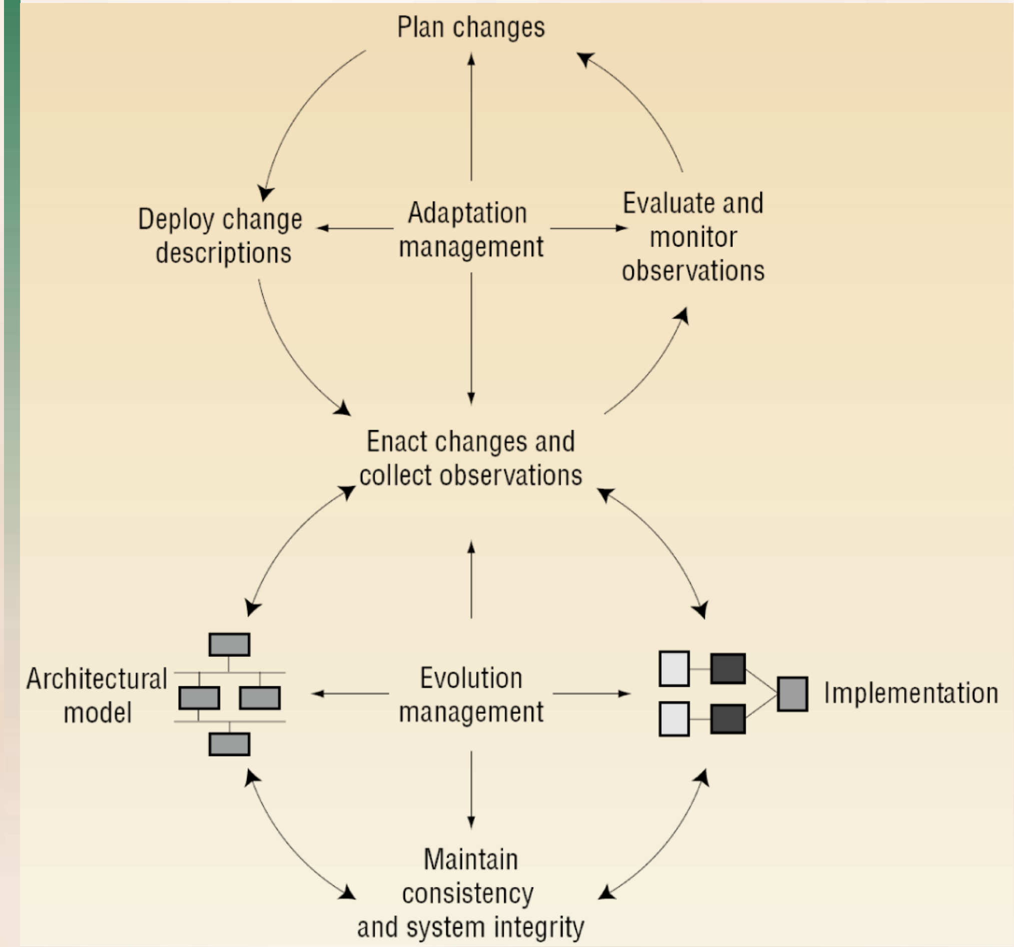
Fundamental Question

- “What is architecture useful for?”
  - Some answers:
    - ...only good documentation.
    - ...only useful design tool.
    - ...only basis for analyses.
- A better answer: A *central* artifact to software systems which is used throughout design, analysis, development, deployment, maintenance, and evolution.

Basic Rules of the Game

- Separation of concerns
- Highly-complex set of activities with different...
  - ...architectural aspects.
  - ...groups of people.
  - ...methods and tools.
- Architecture is the central artifact and *integration* point for these activities.

Specific Example:
Software Evolution and Adaptation

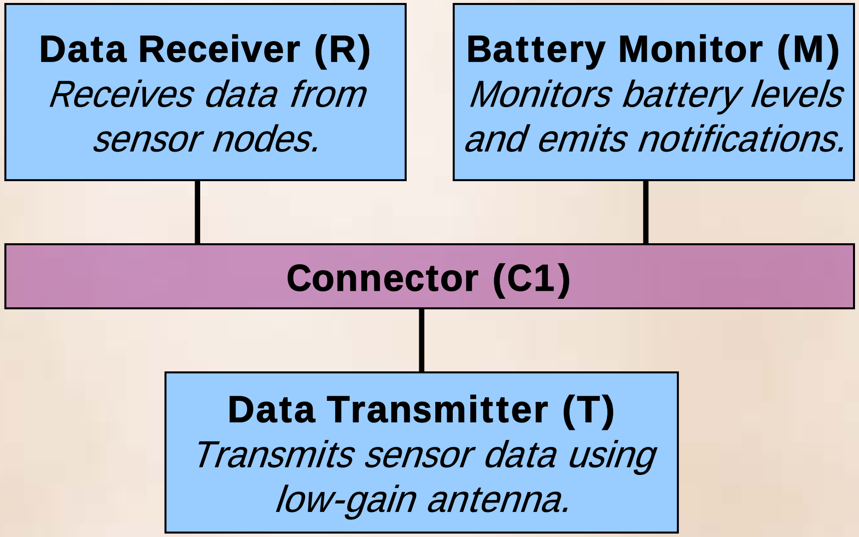

Plan changes
Deploy change descriptions
Adaptation management
Evaluate and monitor observations
Enact changes and collect observations
Architectural model
Evolution management
Implementation
Maintain consistency and system integrity

- Evolution *performed through* the architecture.
- Adaptation decision-making *centered on* the architecture.
- “The architecture is the system.”

Self-Adaptive Systems

- Systems which *autonomously* change in response to dynamic conditions:
  - Behavior, property, environment, etc.
- *Architecture-based* self-adaptive software centers both decision-making and enactment on *explicit* architectural models.

Example:
Sensor Network Re-Transmission


Data Receiver (R)
Receives data from sensor nodes.
Battery Monitor (M)
Monitors battery levels and emits notifications.
Connector (C1)
Data Transmitter (T)
Transmits sensor data using low-gain antenna.

- Component- and event-based systems.
- Longer-range transmission proxy node.
- Balance timeliness with longevity.
  - Continuous vs. burst.

- Some techniques:
  - Adaptation logic built-in to *Transmitter* code.
  - System-specific reconfiguration scripts.

Adaptation Policy Challenges

- *Coupling*
  - Tightly coupled to specific software components.
    - Expressed as part of component logic.
    - Independent of components, but customized.
  - Tightly coupled to specific architectural topologies.
    - Use of application-specific artifacts.
- *Static*
  - Usually pre-specified at design-time.
    - Limited to architect foresight.
  - Difficult to modify during system runtime.
    - Addition of new self-adaptive behavior.

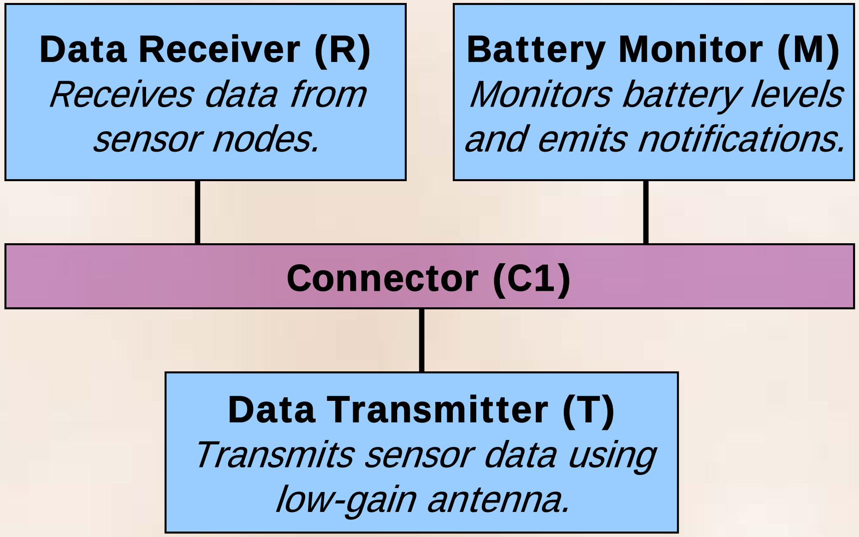
Approach: Architectural Adaptation Policies

- *Rule-based* policy language.
- A mapping from *observations* to *responses*.
    - Responses modify system structure.

```
AdaptationPolicy id
  (Description desc)?
  (Observation id arg*)+
  (Response id arg*)+
```

- *First-class* architectural elements.
    - Explicitly specified at the architectural level.
    - Decoupled from system components.

Example:
Sensor Network Re-Transmission



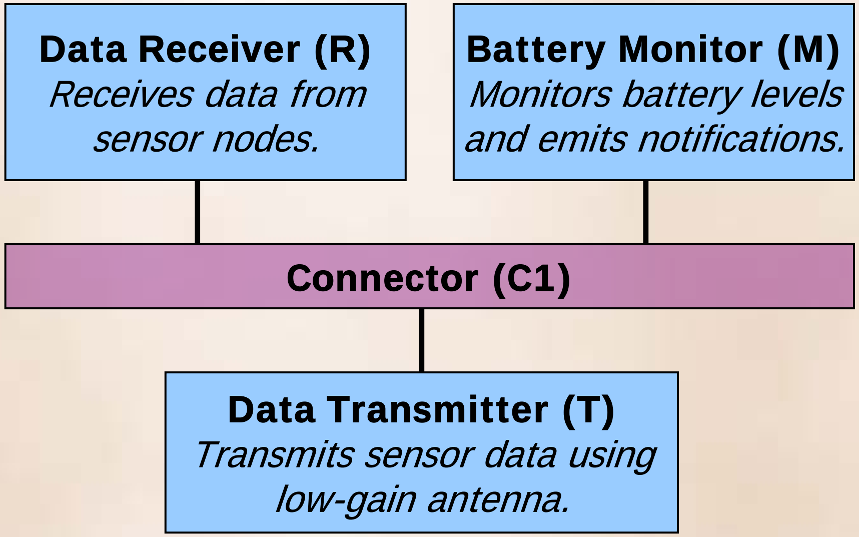
Data Receiver (R)
Receives data from sensor nodes.

Battery Monitor (M)
Monitors battery levels and emits notifications.

Connector (C1)

Data Transmitter (T)
Transmits sensor data using low-gain antenna.

Example:
Sensor Network Re-Transmission



Data Receiver (R)
Receives data from sensor nodes.

Battery Monitor (M)
Monitors battery levels and emits notifications.

Connector (C1)

Data Transmitter (T)
Transmits sensor data using low-gain antenna.

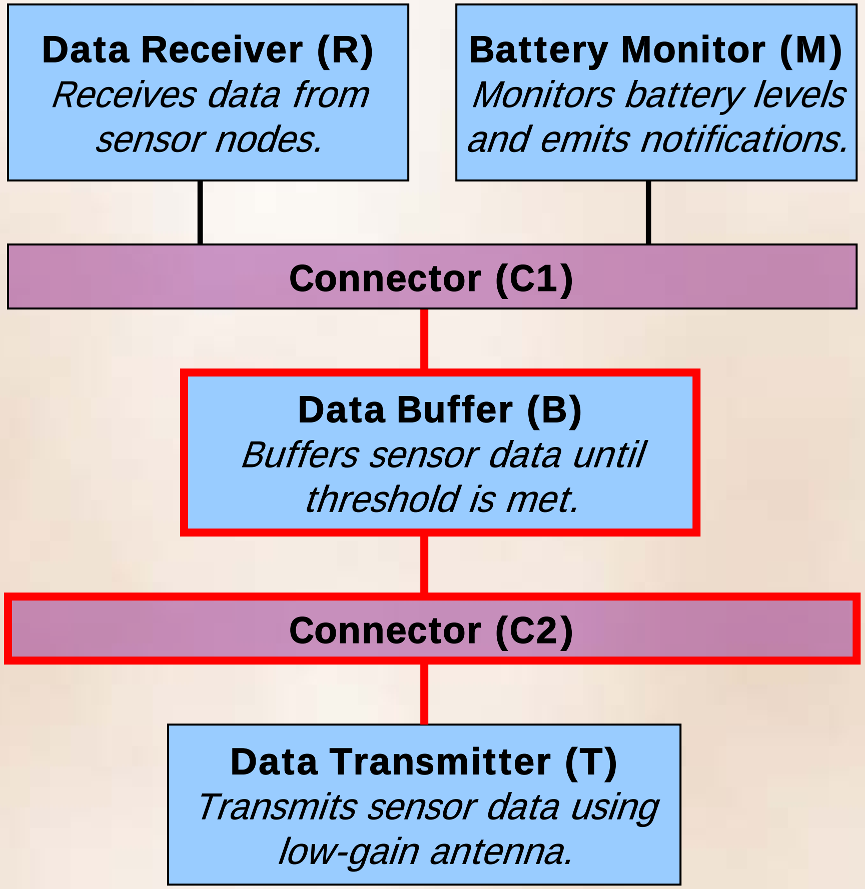
Adaptation Policy

```
AdaptationPolicy switch_burst
  Observation LowBattery
  Response AddComponent(B)
  Response AddConnector(C2)
  Response RemoveLink(C1, T)
  Response AddLink(C1, B)
  Response AddLink(B, C2)
  Response AddLink(C2, T)
```

Example:
Sensor Network Re-Transmission

Adaptation Policy

```
AdaptationPolicy switch_burst
  Observation LowBattery
  Response AddComponent(B)
  Response AddConnector(C2)
  Response RemoveLink(C1, T)
  Response AddLink(C1, B)
  Response AddLink(B, C2)
  Response AddLink(C2, T)
```



Data Receiver (R)
Receives data from sensor nodes.

Battery Monitor (M)
Monitors battery levels and emits notifications.

Connector (C1)

Data Buffer (B)
Buffers sensor data until threshold is met.

Connector (C2)

Data Transmitter (T)
Transmits sensor data using low-gain antenna.

Conclusion

- A strong commitment to architecture as a *central* artifact enables:
  - Clean separation of concerns.
  - Easier integration of heterogeneous tools and methods.
- The architecture is the system.
- Dynamic system evolution and adaptation *through* the architectural model.
  - Use adaptation policies as first-class architectural elements.
- Architecture enables:
  - Decoupling from components and systems.
  - Dynamic, runtime modification of adaptive behavior.
    - “On the prowl” for (self-)adaptive system examples and validation domains.
- *Reuse* of the architectural notations, methods, and tools.