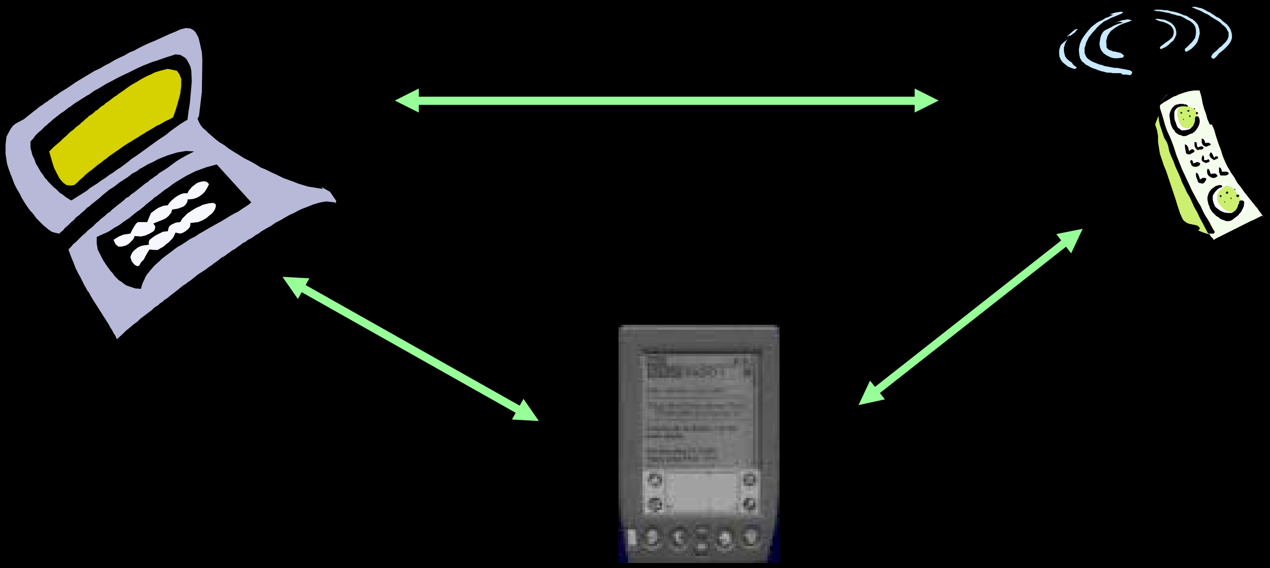
Architecture-Centric Programming for Adaptive Systems

Jonathan Aldrich
Vibha Sazawal
Craig Chambers
David Notkin

University of Washington

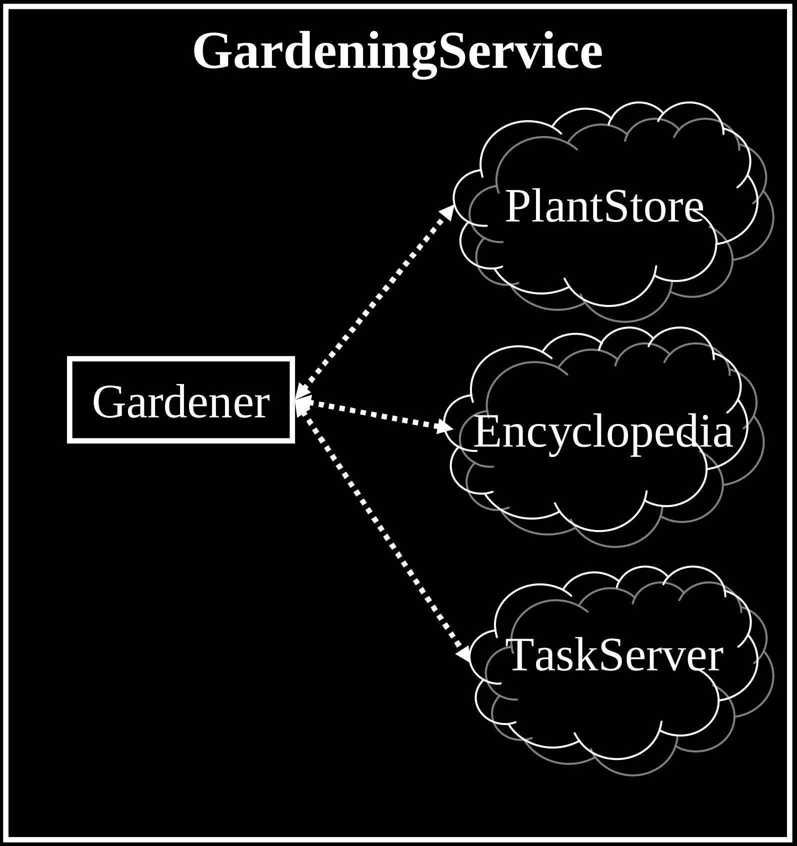
Ubiquitous Computing



- Collaborating embedded devices
- Important class of self-healing systems
  - Frequent change, failures

Motivating Application: PlantCare

- Sensors and robots
  - Care for houseplants
- GardeningService
  - Execution cycle
    - Queries plant moisture
    - Queries encyclopedia
    - Creates watering tasks
  - Self-healing strategy
    - Reconnect to services
    - Restart cycle



Contribution

- Many specific techniques in WOSS
  - Architecture-based adaptation
  - Fault tolerance
  - Self-healing algorithms
- Our contribution: general language support
  - Showing interfaces and connectivity
  - Building adaptable connections
  - Checking properties
  - Separating logic and communication

Our Approach

- ArchJava
  - Adds architecture specification to Java
  - Guarantees communication integrity through types
- Key features
  - Architectural specification
    - Shows interfaces and connectivity
  - User-defined connectors
    - Allow adaptive communication
    - Support rich static typechecking
  - Separates logic from communication

Interfaces and Connectivity

```
component class Gardener {
  port interface PlantInfo {
    requires void statusQuery();
    provides void statusReply(PlantInfo plants[]);
  }
```

- Architecture description language within Java
  - Interfaces

Interfaces and Connectivity

```
component class Gardener {
  port interface PlantInfo {
    requires void statusQuery();
    provides void statusReply(PlantInfo plants[]);
  }
component class GardeningService {
  connect pattern Gardener.PlantInfo,
                  PlantStore.PlantInfo;
```

- Architecture description language within Java
  - Interfaces
  - Connectivity

User-Defined Connectors

```
component class GardeningService {
  connect pattern Gardener.Info,
                PlantStore.Info;
           with RainConnector…
```

- Example: RainConnector
  - Protocol used by PlantCare services
  - Asynchronous XML messages over HTTP

User-Defined Connectors

```
component class GardeningService {
  connect pattern Gardener.Info,
                PlantStore.Info;
           with RainConnector…
```

- Example: RainConnector
  - Protocol used by PlantCare services
  - Asynchronous XML messages over HTTP
- Supports rich connector semantics
  - Adapt to failure
  - Incorporate probes

Static Checking

- Connectors define their own typechecking
  - – Can use Java’s default
  - – Override typecheck() function for custom checks

Static Checking

- Connectors define their own typechecking
  - Can use Java’s default
  - Override typecheck() function for custom checks
- RainConnector
  - Methods return **`void`** (due to asynchrony)
  - Uses structural subtyping

Static Checking

- Connectors define their own typechecking
  - Can use Java’s default
  - Override typecheck() function for custom checks
- RainConnector
  - Methods return `void` (due to asynchrony)
  - Uses structural subtyping
- Other semantics possible
  - Could adapt one type to another
  - Could require meta-information from sender

Separation of Concerns

```
// message sending code
plantInfo.statusQuery();
```

- Services communicate by calling methods

Separation of Concerns

```
// message sending code
plantInfo.statusQuery();

// communication code
public class RainConnector extends Connector ...
```

- Services communicate by calling methods
- Semantics defined by RainConnector

Separation of Concerns

```
// message sending code
plantInfo.statusQuery();

// communication code
public class RainConnector extends Connector ...

// binds PlantInfo using RainConnector
connect pattern Gardener.PlantInfo,
              PlantStore.PlantInfo;
         with RainConnector…
```

- Services communicate by calling methods
- Semantics defined by RainConnector
- Architecture specifies the binding

Previous Work

- Custom UniCon connectors [Shaw *et al.*]
  - Require changing compiler

Previous Work

- Custom UniCon connectors [Shaw *et al.*]
  - Require changing compiler
- Off-the-shelf infrastructures
  - RMI, CORBA, COM
  - Used in C2 connectors [Dashofy *et al.*]
  - Fixed semantics (but see OpenORB)

Previous Work

- Custom UniCon connectors [Shaw *et al.*]
  - Require changing compiler
- Off-the-shelf infrastructures
  - RMI, CORBA, COM
  - Used in C2 connectors [Dashofy *et al.*]
  - Fixed semantics (but see OpenORB)
- Other connector work
  - Focused on semantics, not implementation

Conclusion

- ArchJava language
  - Integrates architecture into implementation
  - Provides user-defined connectors
  - Statically checks architectural integrity
- Reaction from PlantCare developers
  - Understood ArchJava syntax
  - Saw engineering benefits
  - Considering ArchJava in a future system
- Prototype implementation

http://www.archjava.org/