Design Tactics
Relationships Between Tactics and Patterns

- **A pattern** is a high level structural design choice
- **A tactic** is a lower level *design decision* to achieve a QA response
- **Patterns are built from tactics;** if a pattern is a crystal, a tactic is a molecule or an atom.
- For example the layered pattern utilizes the tactics:
  - Increase semantic coherence
  - Encapsulation
  - Abstract common services
  - Restrict communication paths
  - Use an intermediary
Tactics Augment Patterns

- **Patterns solve a specific problem** but are neutral or have **weaknesses** with respect to other qualities.

- E.g., the broker pattern may have:
  - Performance bottlenecks
  - Single point of failure - **availability**
  - Testability complexity in multi-process/processor systems
  - No implicit support for security

---

Consider possible availability tactics
Example – Tactics for Availability
Fault Detection

- Heartbeat - I am alive!
- Ping/Echo - Are you alive?
- Exception - it died!
Heartbeat Tactic Class Model

- **HeartbeatSender** sends a periodic heartbeat message via `pitApat()`
- **HeartbeatReceiver** updates `lastUpdatedTime` with `updateTime()`
- **HeartbeatReceiver** every `checkingInterval` compares the `lastUpdatedTime` against `expireTime` using `checkAlive()`
- If the test fails an exception is thrown
Ping/Echo Tactic Class Model

- **PingSender** sends a ping message at specified `timeIntervals`
- **PingSender** waits for an echo from a ping receiver until `maxWaitingTime`
- If an echo is not received an exception occurs, and it is detected by the fault monitor.
Tactics and QA Interactions (Tradeoffs)

- **Each tactic** has **pluses** (its reason for being) and minuses – **side effects**.
- **Use other tactics** to help alleviate the minuses.
- But nothing is free…
For an availability QA, a common tactic for detecting faults is Ping/Echo. Common side-effects of Ping/Echo are:

- **Security**: how to prevent a ping flood attack?
- **Performance**: how to ensure that the performance overhead of ping/echo is small?
- **Modifiability**: how to add ping/echo to the existing architecture?
Tactics and Interactions - 2

System

Ping/Echo

Add to system  Ping flood  Performance overhead

*
A tactic to address the performance side-effect is “Increase Available Resources”.

Common side effects of Increase Available Resources are:

- **Cost**: increased resources cost more
- **Performance**: how to utilize the increase resources efficiently?
Tactics and Interactions - 4

System

Ping/Echo

Add to system
Ping flood
Performance overhead

Increase Available Resources

Cost
Resource Utilization

*
Tactics and Interactions - 5

A tactic to address the efficient use of resources side-effect is “Scheduling Policy”.
Common side effects of Scheduling Policy are:

- **Modifiability: how to add** the scheduling policy to the existing architecture
- **Modifiability: how to change** the scheduling policy in the future?
Tactics and Interactions – 6

System

Ping/Echo

Add to system
Ping flood
Performance overhead

Increase Available Resources

Cost
Resource Utilization

Scheduling Policy

* Add to system
  Modify policy
A tactic to address the addition of the scheduler to the system is “Use an Intermediary.”

Common side effects of Use an Intermediary are:

- **Modifiability**: how to ensure that all communication passes through the intermediary?
Tactics and Interactions - 8

System

Ping/Echo

Add to system

Ping flood

Performance overhead

Increase Available Resources

Cost

Resource Utilization

Scheduling Policy

Add to system

Modify policy

Use an Intermediary

Ensure usage
Tactics and Interactions – 9

A tactic to address the concern that all communication passes through the intermediary is “Restrict Communication Paths”.

Common side effects of Restrict Communication Paths are:

- **Performance**: how to ensure that the performance overhead of the intermediary are not excessive?

Note: this design problem has now become recursive!
How Does This Process End?

- Each use of tactic introduces **new concerns**.
- Each **new concern** causes **new tactics** to be added.
- Are we in an infinite progression?
- No. Eventually the **side-effects** of each tactic become **small enough to ignore**.
Summary

- An architectural pattern
  - is a package of design decisions that is found repeatedly in practice,
  - has known properties that permit reuse, and
  - describes a class of architectures.

- Tactics are simpler than patterns

- Patterns are underspecified with respect to real systems so they have to be augmented with tactics.
  - Augmentation ends when requirements for a specific system are satisfied.