A Reconfigurable Component Model Using Reflection

JAMES HAWTHORNE, RICHARD ANTHONY

The University of Greenwich, London

Content

- Problems needing to be addressed
- Motivation and Goals
- Component model structure and behaviour
- New and existing project incorporation
- Domains that can benefit
- Relation to resilience
- Example
- Future work
- Conclusions

Motivation

- Inflexible static software
- Autonomic systems need to adapt themselves so a runtime adaptable backbone is necessary
- A framework to enforce the image of a system as a collection of collaborating components to encourage better structured systems.
- A framework to hide unnecessary elements from the designer

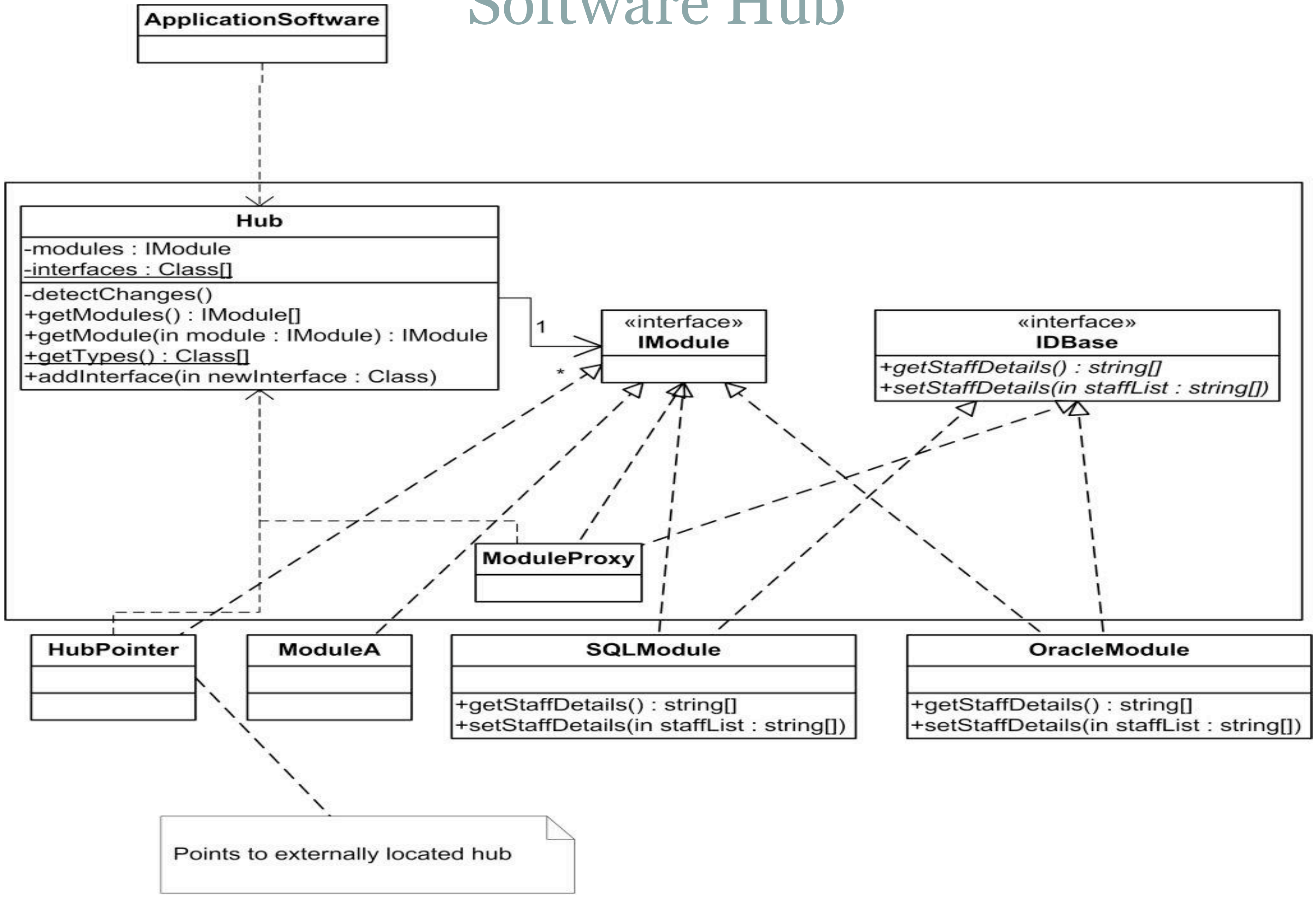
Goals

- Treat software components like hot swappable USB devices
- Make it simple to design runtime adaptable software
- Make it easy to adopt the new style of software design
- Must be possible to design offline software with the possibility of being runtime adaptable in the future
- Must be possible to incorporate legacy software

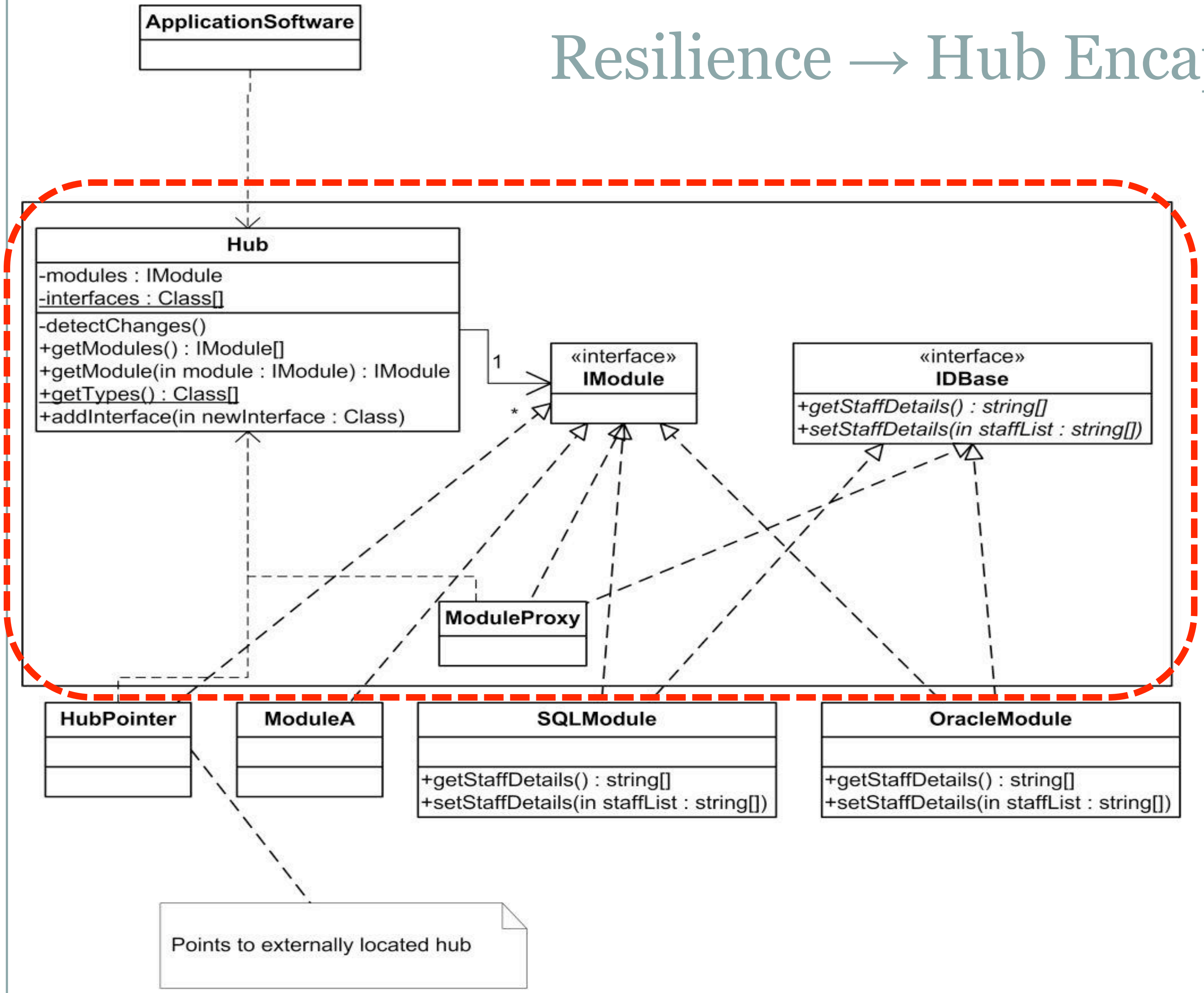
Proposed Approach

- Encapsulate common, complex and non application-specific code into a framework model
- Use interfaces to define objects as types to encourage cohesion and code separation
- Use reflective techniques to detect and utilise new behaviour at runtime
- Use a design which is extensible
  - Extensible applications
  - Extensible framework
  - Extensible distributed software through hierarchical composite pattern integration
- Write governing software which will automatically utilize known types, enabling self-configuring capabilities

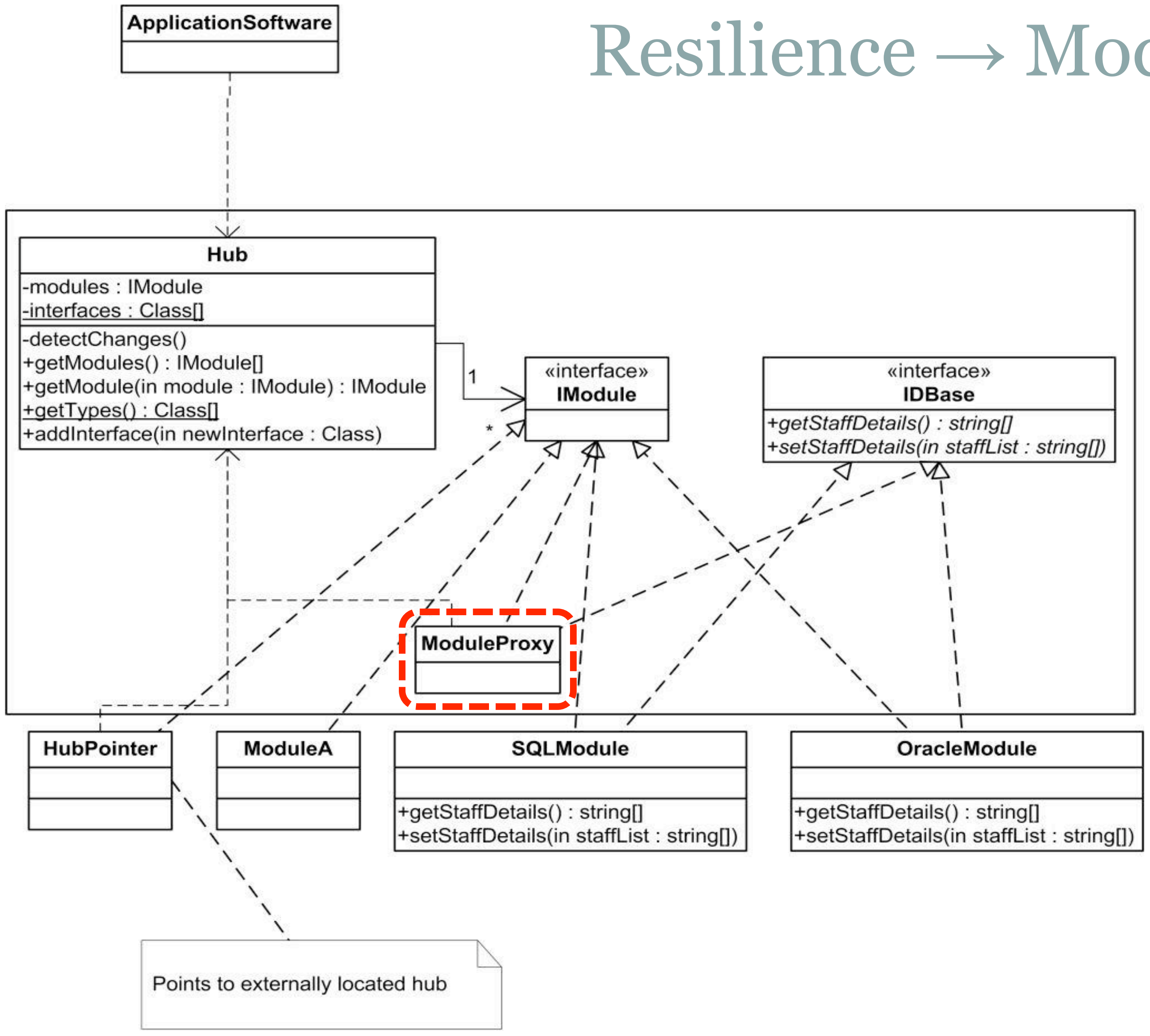
Software Hub


ApplicationSoftware
Hub
-modules : IModule
-interfaces : Class[]
-detectChanges()
+getModules() : IModule[]
+getModule(in module : IModule) : IModule
+getTypes() : Class[]
+addInterface(in newInterface : Class)
1
*
«interface»
IModule
«interface»
IDBase
+getStaffDetails() : string[]
+setStaffDetails(in staffList : string[])
ModuleProxy
HubPointer
ModuleA
SQLModule
+getStaffDetails() : string[]
+setStaffDetails(in staffList : string[])
OracleModule
+getStaffDetails() : string[]
+setStaffDetails(in staffList : string[])
Points to externally located hub

Resilience → Hub Encapsulation


ApplicationSoftware
Hub
-modules : IModule
-interfaces : Class[]
-detectChanges()
+getModules() : IModule[]
+getModule(in module : IModule) : IModule
+getTypes() : Class[]
+addInterface(in newInterface : Class)
1
*
«interface»
IModule
«interface»
IDBase
+getStaffDetails() : string[]
+setStaffDetails(in staffList : string[])
ModuleProxy
HubPointer
ModuleA
SQLModule
+getStaffDetails() : string[]
+setStaffDetails(in staffList : string[])
OracleModule
+getStaffDetails() : string[]
+setStaffDetails(in staffList : string[])
Points to externally located hub

Resilience → Module Proxy


ApplicationSoftware
Hub
-modules : IModule
-interfaces : Class[]
-detectChanges()
+getModules() : IModule[]
+getModule(in module : IModule) : IModule
+getTypes() : Class[]
+addInterface(in newInterface : Class)
1
*
«interface»
IModule
«interface»
IDBase
+getStaffDetails() : string[]
+setStaffDetails(in staffList : string[])
ModuleProxy
HubPointer
ModuleA
SQLModule
+getStaffDetails() : string[]
+setStaffDetails(in staffList : string[])
OracleModule
+getStaffDetails() : string[]
+setStaffDetails(in staffList : string[])
Points to externally located hub

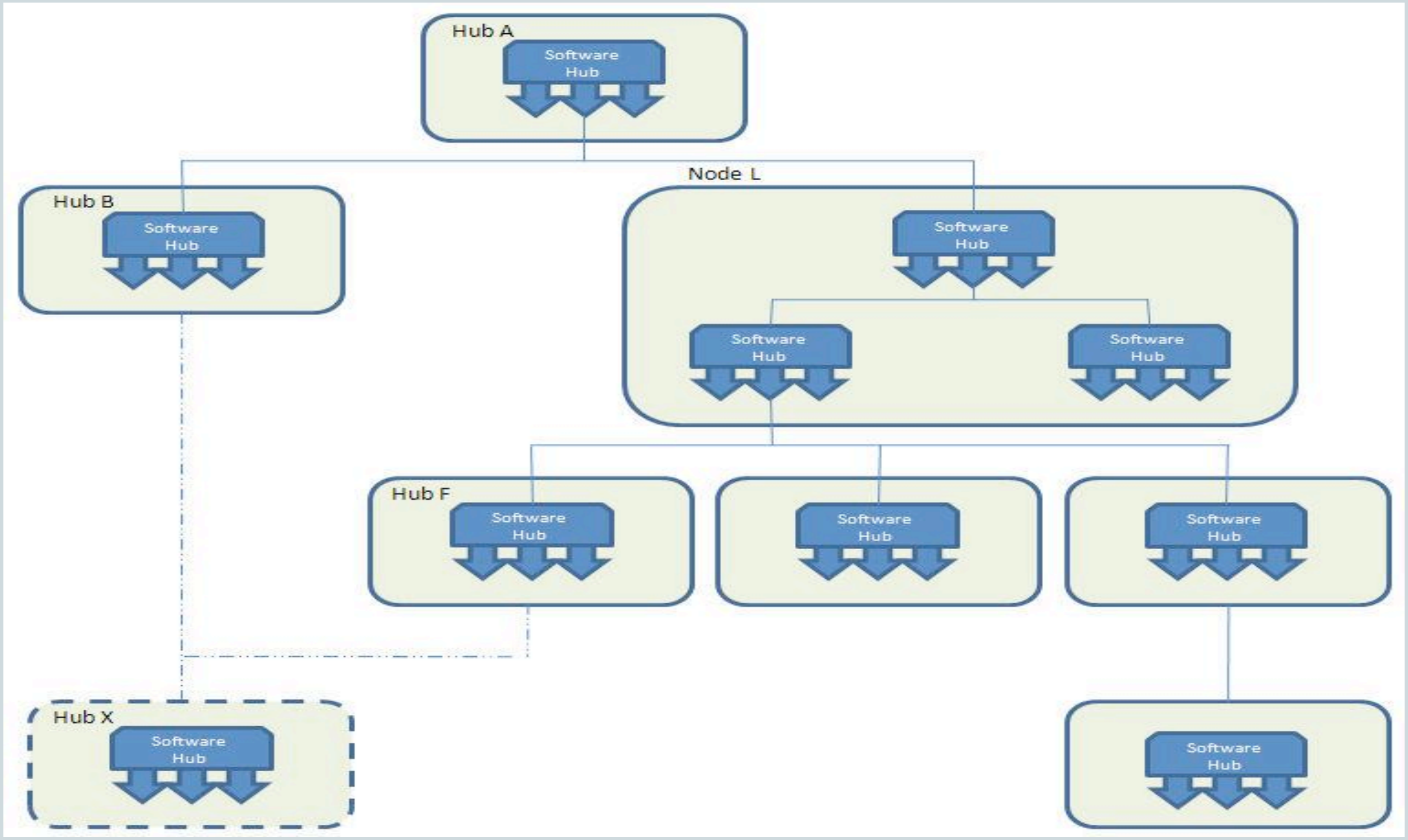
Extensions

- Heterogeneity
- Increase runtime flexibility by making the application software as dynamic as the modules
- Gang of Four design patterns style catalogue of different strategies
- Develop different testing techniques to be built into the hub

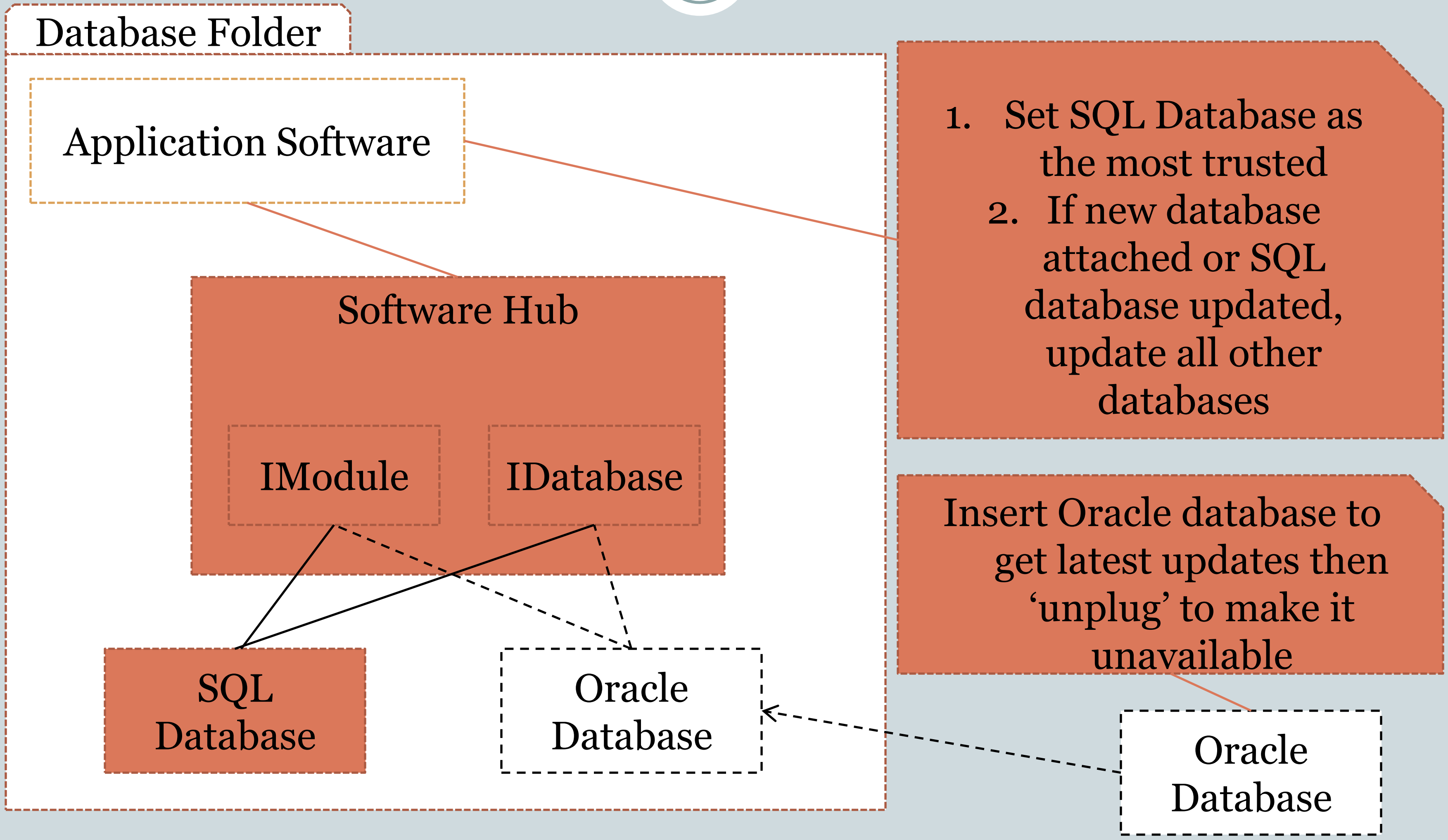
Beneficiaries

- Autonomic systems
- Mission and safety critical systems where halting a system is not an option
- Distributed systems
- Systems which need to incorporate legacy code as part of a larger system
- Systems where connecting and disconnecting specific code regularly would be beneficial. Such as code controlling access to sensitive information

Application to Distributed Systems



Example



1. Set SQL Database as the most trusted
2. If new database attached or SQL database updated, update all other databases

Insert Oracle database to get latest updates then 'unplug' to make it unavailable

Conclusions

- Future software needs such a system. Changes and new requirements inevitably happen and it is not always desirable to halt a system
- Future software will need to manage itself to reduce costs of maintenance
- Runtime adaptations need to be guided and constrained in order to maintain structure and readability
- Component software and reflective techniques need not be two conflicting methods
- Resilience from errors and non-application specific functions should be hidden from the system designer