Feedback Control Architectural Styles Catalog

Sandro Santos Andrade and Raimundo J. de A. Macêdo

Federal University of Bahia - Department of Computer Science

Distributed Systems Laboratory (LaSiD)

{sandros, macedo}@ufba.br

1 A Catalog of Feedback Control-Supporting Architectural Styles

Over the past years an ever increasing number of alternative control techniques has been proposed and successfully applied by control engineers to regulate, compensate disturbances, and optimize systems with different characteristics regarding their dynamics and environmental predictability. Even the basic techniques - such as fixed-gain PID control and derivaties - are still not totally explored and applied by software engineers to enhance adaptability of current software systems. The wide range of control decisions that must be taken, the inexistence of investigation about the suitability of such techniques to control software-intensive systems, and presentation of such approaches in an idiom already accepted by software engineering community, further prevent the comprehensive use of feedback control in modern software systems.

1.1 Feedback Control Design Dimensions

When designing self-adaptive systems software engineers should make choices about a number of aspects (design dimensions) related to feedback control. In order to set the stage for deriving a catalog of architectural styles of feedback control we should determine such dimensions, its possible values, and consequent impacts of setting a given value to a specific dimension. Table 1 presents the control design dimensions that have influenced the derivation of the architectural styles presented in the proposed catalog.

To enable critical thinking and allow comparison between different architectural choices, the proposed architectural styles will be documented by a rubric, describing its summary, involved components and connectors, expected topologies, constraints imposed by the style, qualities yielded, typical used, and cautions.

1.2 The Catalog

The following architectural styles regarding feedback control have been found in several implementation of self-managed systems, with diverse purposes, architectures, dynamics, and constraints.

Dimension	Sub-Dimension	Values
Control goal		Regulation Disturbance Rejection Optimization
Control feed		Open-loop Closed-loop Open-loop and Closed-loop
System model formalism		First principles Black-box
Control adaptiveness	Fixed gain Adaptive	Gain scheduling MIAC MRAC
Control loop dimension		SISO Multiple SISO MIMO
Control law		PID and derivatives Stochastic Reconfiguration
Control reconfiguration		Static (non-reconfigurable) Reconfigurable
Control schema		Single loop Cascaded (nested) Hierarchical Decentralized Hybrid
Control triggering		Time-triggered Event-triggered Hybrid

Table 1: Design dimensions for feedback control-based self-managed systems

1.2.1 #1: Feedforward Control - (FFC)

Summary	A separate component (controller) measures disturbance input and acts accordingly to drive the system to the expected output
Components	System, controller, effector, disturbance measurer
Connectors	(Remote) procedure call, event bus or data access
Data Elements	Reference value(s), input values, disturbance values
Topology	Linear with two entry points: (disturbance measurer/reference input -> controller -> system)
Variants	None
Qualities yielded	Proactive behaviour; no need for output measuring
Typical uses	When system modeling is quite complicated, disturbance modeling is available, control input is a deterministic function of reference input, and the target system is stable
Cautions	When disturbance modeling is complicated; it cannot make stable an unstable system; it does not adapt to unmodeled disturbances; it needs an accurate system model
Relations to programming languages or environments	None

1.2.2 #2: P(ID) Feedback Control with System Identification - (PID-SI)

Summary	A separate component (controller) measures system output (feedback) and acts accordingly to drive the system to the expected output
Components	System, controller, effector, sensor, transducer/QoS subsystem (optional, if output not directly delivered by the system itself)
Connectors	(Remote) procedure call, event bus or data access
Data Elements	Reference value(s), input values, output values, transduced values (optional)
Topology	Circular with one entry point: (reference input -> controller -> system -> measured output [-> transducer/QoS subsystem] -> ...)
Variants	#2.1: PID-SI with Precompensation - (PID-SI/PC) #2.2: PID-SI with Sensor Delay - (PID-SI/SD) #2.3: PID-SI with Filtering - (PID-SI/F)
Qualities yielded	Reactive behaviour; adaptation to unmodeled disturbances; no need for an accurate system model; can make stable an unstable system
Typical uses	When a good enough system model is available, disturbance modeling is quite complicated, target system is unstable but linear or with identificable linear operating regions
Cautions	When disturbance spans over a wide range; when system is primary non-linear or have dynamics that are difficult to be modeled; when structural reconfiguration is needed
Relations to programming languages or environments	None

1.2.3 #3: Feedback and Feedforward Control - (FBFFC)

Summary	A separate component (controller) measures disturbance input and system output (feedback) and acts accordingly to drive the system to the expected output
Components	System, controller, effector, disturbance measurer
Connectors	(Remote) procedure call, event bus or data access
Data Elements	Reference value(s), input values, disturbance values, output values, transduced values (optional)
Topology	Circular with two entry points: (disturbance measurer/reference input -> controller -> system -> measured output [-> transducer/QoS subsystem] -> ...)
Variants	None
Qualities yielded	Proactive and reactive behaviour
Typical uses	When a good enough system AND disturbance models are available
Cautions	Both from open-loop only and closed-loop only scenarios
Relations to programming languages or environments	None

1.2.4 #4: Multiple-SISO Feedback Control - (MSISO)

Summary	A separate component (controller) measures several system outputs (feedbacks) and acts accordingly by setting a group of control input variables. One specific input variable impacts only its respective output variable, there is no cross-dependency between inputs and outputs
Components	System, controller, effectors, sensors, transducer/QoS subsystem (optional, if output not directly delivered by the system itself)
Connectors	(Remote) procedure call, event bus or data access
Data Elements	Reference value(s), input values, output values, transduced values (optional)
Topology	Circular with several entry points: (reference inputs -> controller -> system -> measured outputs [-> transducer/QoS subsystem] -> ...)
Variants	None
Qualities yielded	All from PID-SI plus the ability to control several measured outputs
Typical uses	All from PID-SI AND system's service level is achieved only by manipulating a SET of independent control inputs
Cautions	When system presents cross-dependency between control inputs and measured outputs
Relations to programming languages or environments	None

1.2.5 #5: MIMO Feedback Control - (MIMO)

Summary	A separate component (controller) measures several system outputs (feedbacks) and acts accordingly by setting a group of control input variables. One specific input variable impacts more the one output variable (cross-dependency between inputs and outputs)
Components	System, controller, effectors, sensors, transducer/QoS subsystem (optional, if output not directly delivered by the system itself)
Connectors	(Remote) procedure call, event bus or data access
Data Elements	Reference value(s)/vector(s), input values/vectors, output values/vectors, transduced values/vectors (optional)
Topology	Circular with several entry points: (reference inputs -> controller -> system -> measured outputs [-> transducer/QoS subsystem] -> ...)
Variants	#5.1: LQR MIMO Feedback Control - (MIMO/LQR) #5.2: MPC MIMO Feedback Control - (MIMO/MPC) #5.3: State-Space MIMO Feedback Control - (MIMO/SS)
Qualities yielded	All from PID-SI plus higher adaptability to unmodeled disturbances; better controlability
Typical uses	All from PID-SI AND system's service level is achieved only by manipulating a SET of dependent control inputs
Cautions	When interplay between control inputs is quite complicated (therefore requiring higher order models)
Relations to programming languages or environments	None

1.2.6 #6: Adaptive Feedback Control - (AFC)

Summary	A separate component (controller) measures system output (feedback) and acts accordingly to drive the system to the expected output. There is no complete knowledge about the system, its operating regions, linearities, and dynamics
Components	System, controller, effectors, sensors, transducer/QoS subsystem (optional, if output not directly delivered by the system itself), model estimator, controller designer
Connectors	(Remote) procedure call, event bus or data access
Data Elements	Reference value(s), input values, output values, transduced values (optional), system parameters, controller parameters
Topology	One feedback loop with three responsibilities: control the system output (faster time scale), refine the system model, and adapt the controller (slower time scale)
Variants	#6.1: Model Identification Adaptive Feedback Control - (AFC/MIAC) #6.2: Model Reference Adaptive Feedback Control - (AFC/MRAC) #6.3: Feedback Control with Gain Scheduling - (AFC/GS)
Qualities yielded	All from PID-SI plus higher adaptability to unmodeled disturbances, better controlability, and better robustness
Typical uses	When system and disturbance dynamics are unknow AND a single reconfigurable control strategy (law) is suitable
Cautions	When adaptation overhead is prohibitive (if chattering is present); when a single control strategy (law) is not enough
Relations to programming languages or environments	None

1.2.7 #7: Cascated (nested) Feedback Control - (CFBC)

Summary	A separate component (controller) measures system output (feedback) and acts accordingly to drive the system to the expected output, but this expected output is not complete known in advance and, therefore, must be also controlled.
Components	System, inner controller, system and inner controller effectors, system sensors, transducer/QoS subsystem (optional, if output not directly delivered by the system itself), outer controller
Connectors	(Remote) procedure call, event bus or data access
Data Elements	Goal(s), reference value(s), inner control input values, system input values, output values, transduced values (optional)
Topology	Two feedback loops with different time scales: an inner feedback loop acting on the system (faster time scale) and an outer feedback loop acting on the inner controller (slower time scale)
Variants	None
Qualities yielded	All from PID-SI plus higher level goal specification
Typical uses	All from PID-SI AND the reference input changes accordingly to some higher level goal
Cautions	The inner and outer controller should operate on different and compatible time scales. Otherwise, unexpected behavior could emerge as a result of overreaction to system output mainly if its has a proeminent stochastic component
Relations to programming languages or environments	None

1.2.8 #8: Hierarchical Feedback Control - (HFC)

Summary	System goals are described by a set of reference inputs on different abstraction levels. Controllability flows downwards (topdown) through an hierarchy of controllers. Cascated (nested) control is a particular case of hierarchical feedback control
Components	Subsystems, controllers, system and controllers effectors, system sensors, transducers/QoS subsystem
Connectors	(Remote) procedure call, event bus or data access
Data Elements	Goal(s), reference value(s), control input values, subsystems input values, output values, transduced values (optional)
Topology	Hierarchical set of controllers with central control authority
Variants	None
Qualities yielded	Even higher level goal specification; suitable for distributed systems scenarios
Typical uses	All from PID-SI AND the high level goals can be modeled as a set of reference input at different levels
Cautions	Controllers in a higher level should operate at a lower time scale than controller in a lower level. Overreaction may be a problem. It may suffer from scalability issues. Inadequate when there is no single control authority
Relations to programming languages or environments	None

1.2.9 #9: Decentralized Feedback Control - (DFC)

Summary	A set of controlled subsystems adapt their own behaviour without any knowledge about global system state. Communication between subsystems is performed as necessary. System properties emerge from the interplay of locally-controlled subsystems
Components	System, controller, effectors, sensors, transducer/QoS subsystem (optional, if output not directly delivered by the system itself), communication/middleware infrastructure
Connectors	(Remote) procedure call, event bus, arbitrator or data access
Data Elements	Reference value(s), input values, output values, transduced values (optional), system parameters/topologies, controller parameters/structure, database queries/insertions, reconfiguration messages
Topology	Can vary arbitrarily
Variants	#9.1: Hybrid Hierarchical/Decentralized Control - (HHD)
Qualities yielded	Emergent behaviour, self-organizability, high availability, high scalability
Typical uses	When scalability is a primary concern, when there is no central control authority
Cautions	When complex intra-controller interaction is required, when temporal predictability of system's goal achievement is required
Relations to programming languages or environments	None

1.2.10 #10: Reconfigurable Control - (RC)

Summary	A separate component (controller) measures system output (feedback) and acts accordingly to drive the system to the expected output. There is neither complete knowledge about the system, its operating regions, linearities, and dynamics; nor a single control strategy (law) that performs well in all expected scenarios. Achieving the system's goals may also require changing the system run-time architectural structure
Components	System, controller, effectors, sensors, transducer/QoS subsystem (optional, if output not directly delivered by the system itself), evaluator, controller designer, reconfigurer, model database, controller database
Connectors	(Remote) procedure call, event bus, arbitrator or data access
Data Elements	Reference value(s), input values, disturbance values
Topology	One feedback loop with multiple responsibilities: control the system output (faster time scale), refine the system model, adapt the controller (slower time scale), decide when a system/controller restructuring is needed, query/update the model e controller databases, enact system and controller structural changes
Variants	None
Qualities yielded	All from AFC plus an even higher adaptability to unmodeled disturbances, better controlability, and better robustness
Typical uses	When system and disturbance dynamics are unknow AND a single reconfigurable control strategy (law) is NOT suitable, when system (plant) structural reconfiguration is required in order to achieve system's goals
Cautions	A more sophisticated infrastructure for component discovery and run-time component assembly/evaluation if required, its high overhead may be prohibitive in real-time scenarios
Relations to programming languages or environments	None

1.2.11 #11: Intelligent Control - (IC)

Summary	The controller uses heuristics and/or learning algorithms to control the system, usually described by an stochastic model
Components	System, controller, effectors, sensors, transducer/QoS subsystem (optional, if output not directly delivered by the system itself), inference engine/statistical classifier/(de)fuzzifier
Connectors	(Remote) procedure call, event bus, arbitrator or data access
Data Elements	Reference value(s), input values, output values, transduced values (optional), rules, facts, queries
Topology	Can vary arbitratily
Variants	None
Qualities yielded	High generality
Typical uses	When system modeling is quite complicated (because of inherent non-linearities)
Cautions	I may require some learning time to present good controllability; conflicts in rule processing
Relations to programming languages or environments	None