

ANIMA
Internet-Draft
Intended status: Informational
Expires: October 23, 2016

J. Strassner
Huawei Technologies
J. Halpern
Ericsson
M. Behringer
Cisco Systems
April 19, 2016

The Use of Control Loops in Autonomic Networking
draft-strassner-anima-control-loops-01

Abstract

This document defines the requirements for an autonomic control loop, describes different types of control loops, and explains how control loops are used in an Autonomic System. Control loops are used to enable Autonomic Network Management systems to adapt the behavior of the systems that they manage to respond to changes in user needs, business goals, and/or environmental conditions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 23, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
2.	Conventions Used in This Document	4
3.	Terminology	4
3.1.	Acronyms	4
3.2.	Definitions	4
3.2.1.	Control Loop	4
3.2.2.	Control Loop, Open	4
3.2.3.	Control Loop, Closed	5
3.2.4.	Control Loop, Proportional	5
3.2.5.	Control Loop, Proportional-Derivative	5
3.2.6.	Control Loop, Proportional-Integral-Derivative (PID) ..	5
3.2.7.	Control Loop, Cascade	5
3.2.8.	Control System	5
4.	Requirements for Control Loops in Autonomic Networks	5
4.1.	Mandatory Autonomic Control Loop Requirements	6
4.1.1.	Observe and Collect Data	6
4.1.2.	Orient Data	6
4.1.3.	Analyze Data	7
4.1.4.	Plan Actions Based on Oriented Data	7
4.1.5.	Decide Which Plan(s) to Execute	7
4.1.6.	Execute the Plan(s)	7
4.1.7.	Detect and Resolve Conflicts	8
4.2.	Desired Autonomic Control Loop Requirements	8
4.2.1.	Observe and Collect Data From External Systems	9
4.2.2.	Orient Data from External Systems	9
4.2.3.	Execute One or More Machine Learning Algorithms	9
4.2.4.	Register Control Loop Capabilities	10
4.2.5.	Register Control Loop Requirements	10
4.3.	Optional Autonomic Control Loop Requirements	10
4.3.1.	Use of A Single Information Model	11
4.3.2.	Use of Ontologies	11
4.3.3.	Collaborate With Other Control Loops	11
5.	Control Loop Usage in Autonomic Networks	12
5.1.	Autonomic Management	12
5.2.	Policy and Context	13
5.3.	Types of Policies	14

5.3.1.	Policies Organized by Actors	14
5.3.2.	Policies Organized by Technology	15
5.4.	Policy Conflicts	15
5.4.1.	Policy Conflicts Caused by Technology	15
5.4.2.	Policy Conflicts Caused by Different Systems	16
5.5.	Control Loops	16
5.5.1.	Types of Control	16
5.5.2.	Types of Control Loops	17
5.5.3.	Management of an Autonomic Control Loop	18

Table of Contents (continued)

6.	Security Considerations	18
7.	IANA Considerations	18
8.	Acknowledgements	18
9.	References	19
	Authors' Addresses	20

[1.](#) Introduction

The document "Autonomic Networking - Definitions and Design Goals" [[RFC7575](#)] explains the fundamental concepts behind Autonomic Networking. In [section 1](#), it says: "The fundamental concept involves eliminating external systems from a system's control loops and closing of control loops within the Autonomic System itself, with the goal of providing the system with self-management capabilities...". In [section 5](#), it also describes a high-level reference model [[draft-ietf-anima-reference-model-01](#)]. This document expands on the definition and use of control loops [[draft-ietf-anima-reference-model-01](#)] ([section 8.5](#)) in Autonomic Systems to self-adapt to various changes to achieve self-management.

In particular, this document describes how control loops are used in Autonomic Network Management to enable the Autonomic System (and its subsystems, which may or may not be autonomic) to adapt (on its own) to enable Autonomic Network Management systems to adapt the behavior of the systems and components that they manage to respond to changes in user needs, business goals, and/or environmental conditions.

Such changes can alter the goals that the Autonomic System must achieve, or how those goals are achieved. For example, this may result in the offering of changed or even new services and resources.

Control loops operate to continuously observe and collect data about the set of managed entities, components, and systems that are being managed, as well as the context in which they are operating. This enables the Autonomic Management System to understand changes in the behavior of the system being managed, analyze those changes, and then provide actions to move the state of the system being managed towards a common goal. Self-adaptive systems move decision-making from static, pre-defined commands to dynamic processes computed at runtime.

This document defines the requirements for an autonomic control loop, describes different types of control loops, and explains how control loops are used in an Autonomic System.

As discussed in [[RFC7575](#)], the goal of this work is not to focus exclusively on fully autonomic nodes or networks. In reality, most networks will run with some autonomic functions, while the rest of the network will not. The reference model defined in [[draft-ietf-anima-reference-model](#)] allows for this hybrid approach.

This is a living document, and will evolve with the technical solutions developed in the ANIMA WG.

[2.](#) Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [[RFC2119](#)] significance.

[3.](#) Terminology

This section defines acronyms, terms, and symbology used in the rest of this document.

[3.1.](#) Acronyms

ANI	Autonomic Network Infrastructure
CLI	Command Line Interface
OAM&P	Operations, Administration, Management, and Provisioning
PID	Proportional-Integral-Derivative (a type of controller)

[3.2.](#) Definitions

This section defines the terminology that is used in this document.

[3.2.1.](#) Control Loop

A control loop is a type of control system that manages the behavior of the devices and systems that it is governing.

[3.2.2.](#) Control Loop, Open

A control loop whose output is generated based only on the input(s) it receives.

Strassner, et al.	Expires October 23, 2016	[Page 4]
-------------------	--------------------------	----------

Internet-Draft	Autonomic Control Loop Usage	Apr 2016
----------------	------------------------------	----------

[3.2.3.](#) Control Loop, Closed

A control loop whose output is a function of the current output and a set of corrections made to that output based on feedback.

[3.2.4.](#) Control Loop, Proportional

A type of control algorithm that generates a stronger response when the system is farther away from its goal state. In other words, the response of the control algorithm is proportion to the amount of error received.

[3.2.5.](#) Control Loop, Proportional-Derivative

A type of control algorithm that uses the rate-of-change of the error with time. In other words, it uses the error when the system is far away from the goal state, and then corrects for the momentum of the system as it gets closer to the goal state.

[3.2.6.](#) Control Loop, Proportional-Integral-Derivative (PID)

A type of control algorithm that adds an integral action to a proportional derivative controller. The integral term eliminates long-term steady-state errors. By integrating the error over time, the controller can drive the system closer to the goal state.

[3.2.7.](#) Control Loop, Cascade

A type of controller where multiple controllers (usually PID) are used to provide fine-grained control. The simplest type of cascade control is two PIDs, where one PID controls the

[3.2.8.](#) Control System

A control system consists of systems and processes that collectively govern the output of the system.

[4.](#) Requirements for Control Loops in Autonomic Networks

The following subsections define the requirements that Autonomic Control Loops MUST, SHOULD, and MAY provide.

[4.1.](#) Mandatory Autonomic Control Loop Requirements

An autonomic control loop MUST be able to perform the following functions as part of its operation:

- o Observe and collect data from the system being managed

- o Orient these data, so that their meaning and significance can be understood in the proper context
- o Analyze the collected data through filtering, correlation, and other mechanisms to define a model of past, current, and future states
- o Plan different actions based on inferring trends, determining root causes, and similar processes
- o Decide which plan(s) to execute, and when
- o Execute the plan(s), and then repeat these steps
- o Detect and resolve any conflicts between different goals that the Autonomic System is given

These seven requirements are further explained in the following subsections.

[4.1.1.](#) Observe and Collect Data

Control loops begin with input data. An autonomic control loop MUST be able to observe and collect data, as instructed by an Autonomic Management System. Without the proper input data, the control loop will be ineffective at best, and likely useless. However, many data in their raw form are not easy to understand by an Autonomic System, and may not be compatible with other data that have been collected by the Autonomic System. Hence, this stage is a mostly passive ability to collect data that is meaningful for the management process, and relies heavily on the next (orientation) step.

[4.1.2.](#) Orient Data

The orientation of data ensures that those data are taken in the correct context. This enables their meaning, as well as their relative importance, to be properly assigned. Autonomic control loops MUST orient data.

Orientation of data was the second step in Boyd's OODA loop [[Boyd95](#)]. OODA stands for Observe, Orient, Decide, and Act. The FOCAL [[Strassner07](#)] control loops are an extension of OODA.

The orientation step of OODA is critical, as it determines how observations, decisions, and actions are performed. This mimics human behavior, since most people react according to how they perceive the world, as opposed to how the world really is.

In FOCAL, the orient step is a model-based translation of received data to normalize those data into a common form, where they can each contribute to the overall perception of the System that is being managed. For example, data from a variety of sensors (e.g., pressure, visual, thermal, etc.) can be fitted into an overall model that also includes performance of IP services, device interfaces, and other entities. Without this normalization of applicable device information, the overall context of the system is not known. This in turn increases the risk of the wrong decision being made.

[4.1.3.](#) Analyze Data

The analysis of data is critical for enabling the control loop to operate properly. Autonomic control loops **MUST** be able to analyze data, after they have been oriented, in order to determine the set of critical properties that the control loop is operating against.

For example, the analysis might derive the current state of the system being managed; this can then be compared to the desired state of the system to define an error function that can be fed back into the control loop. As another example, one or more attributes could be monitored to determine whether the system is operating as planned or not; again, an error function is then defined that can be fed back into the control loop. This step is part of the Orient function in OODA, but is separate in FOCAL.

[4.1.4.](#) Plan Actions Based on Oriented Data

Once the analysis is done, the Autonomic System then understands if its current behavior needs to be modified or changed. This takes the form of one or more plans. An autonomic control loop **MUST** be able to generate one or more plans to govern the behavior of the system being managed. There can be many different ways to solve a problem; a plan is built for each way to enable them to be compared and contrasted.

[4.1.5.](#) Decide Which Plan(s) to Execute

Given a set of plans generated by the control loop, a control loop **MUST** be able to choose which plan, or set of plans, to execute. If multiple plans are to be executed, then the autonomic control loop **MUST** define the order (if any) of execution of each control loop.

In FOCAL, each plan is evaluated with respect to the current context. This enables context to optimize which plan, or set of plans, are best suited to achieving the goal(s) in managing the behavior of the system. Note that this forms another very important loop in FOCAL: context selects policies. As context changes, a new working set of policies are selected. Hence, the behavior of the Autonomic System adapts to changing context.

[4.1.7.](#) Detect and Resolve Conflicts

Autonomic systems typically use policy rules to either help in making decisions, or to provide actions to take as part of the control loop. An Autonomic System MUST be able to detect, and then resolve, conflicts. Both MAPE [[Kephart03](#)] and FOCAL [[Strassner07](#)] provide several examples of this behavior.

[4.2.](#) Desired Autonomic Control Loop Requirements

An autonomic control loop SHOULD be able to perform the following functions as part of its operation:

- o Observe and collect data from other devices and/or systems that can influence the behavior of the system being managed
- o Orient data from other devices and/or systems that can influence the behavior of the system being managed, so that their meaning and significance can be understood in the proper context
- o Execute one or more machine learning algorithms that can learn from and make predictions on monitored data. This enables more efficient adaptivity. It also enables "shortcuts" to be built that enable one or more functional blocks of the control loop to be skipped because the Autonomic System already recognizes what needs to be corrected in the system.
- o Register the capabilities that this control loop can govern with a collection of other Autonomic Systems that it may exchange information and control with

- o Register the requirements that this control loop needs in order to accomplish its tasks

These five requirements are further explained in the following subsections.

[4.2.1.](#) Observe and Collect Data From External Systems

Autonomic Systems are context-aware. This means that the context of the Autonomic System helps determine what actions (if any) should be taken at any given time. Therefore, Autonomic Systems SHOULD take into account data that directly and indirectly affect the goals of the Autonomic System. This includes data that affect the Autonomic System itself and/or data that affect the system that is being governed by the Autonomic System.

Data that directly affects the Autonomic System are data that belong to the Autonomic System, and/or the system being governed by the Autonomic System. Data that indirectly affects the Autonomic System are data that belong to systems that are neither the Autonomic System nor the system that the Autonomic System is managing.

[4.2.2.](#) Orient Data from External Systems

All data, regardless of whether it directly or indirectly affects the Autonomic System, SHOULD be oriented so that a common frame of reference is built to consider the relative importance of observed and collected data. This orientation places ensures that data are compared and analyzed in the correct context. This enables their meaning, as well as their relative importance, to be properly assigned. Autonomic control loops SHOULD orient external data.

[4.2.3.](#) Execute One or More Machine Learning Algorithms

Machine learning refers to algorithms that can learn from, and make predictions about, data. Machine learning algorithms use a model, built from a set of exemplar data, to make predictions and decisions. More formally, [[Mitchell97](#)] defines machine learning as:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Machine learning provides the ability for the Autonomic System to learn from its environment, without burdening the developer to program an explicit set of steps to do so. As such, it is well-suited for providing the basis for learning from the environment in order to adapt the services and resources that it offers in order to maintain, protect, or better fulfil its goals.

[4.2.4.](#) Register Control Loop Capabilities

Autonomic systems provide a number of functional capabilities. Sometimes, a control loop of one Autonomic System may have excess processing available that could be used by other control loops of the same or different Autonomic Systems. Therefore, an autonomic control loop SHOULD register the functional capabilities that it provides, so that other autonomic control loops can request the use of one or more of those functional capabilities. Note that the use of models and/or ontologies greatly simplifies this task, as models and ontologies provide a common vocabulary, complete with meanings, that are shared by all autonomic elements in an Autonomic System.

[4.2.5.](#) Register Control Loop Requirements

Autonomic systems provide a number of functional capabilities. Sometimes, a control loop can benefit from other resources (that are part of other Autonomic Networks) that available to perform one or more of the functions required by the control loop.

This may be because the control loop has run out of resources from its own autonomic elements, or it may be because other autonomic elements can supply more powerful, or robust, versions of the functions that a control loop needs compared to the functions

provided by its own autonomic elements. In order for this to occur, an autonomic control loop SHOULD register its requirements. This enables other autonomic elements to provide resources and/or services to the autonomic control loop, as needed.

[4.3.](#) Optional Autonomic Control Loop Requirements

An autonomic control loop MAY be able to perform the following functions as part of its operation:

- o Use a single information model to help normalize observed and collected data.
- o Use one or more ontologies to define semantics for data. If models define facts, then ontologies conceptually define the semantics for those facts. This is critical in enabling the Autonomic System to reason and learn.
- o Collaborate with other control loops of other Autonomic Systems, so that autonomic control can be extended beyond the confines of any one system to a collection of Autonomic Systems

These three requirements are further explained in the following subsections.

[4.3.1.](#) Use of A Single Information Model

Autonomic Systems MAY use an information model to define common concepts used by all systems that are interacting with each other.

The advantage of using an information model is that it defines a set of concepts in a technology-neutral format. This is important, because most management systems use a variety of different data models (e.g., directories, relational databases, in-memory databases, and others). Each of these data models structures and organizes data differently, and has very different ways of performing basic operations (e.g., create, read, update, and delete) on those data using very different protocols. Hence, if a data object is updated in one data model, how can the system reliably update other instances of that data object if the

protocol, representation, data type, and other elements of the data object are different?

The role of an information model is to define common concepts once; this enables a set of mappings between the information model and each data model to be defined, so that data coherency is maintained in each data model.

[4.3.2.](#) Use of Ontologies

Autonomic Systems MAY use a set of ontologies for defining the meaning associated with different facts collected by the Autonomic System. Facts can be derived from models as well as from the ontologies themselves.

Information and data models are important. However, neither type of model can typically support reasoning, because neither type of model defines formal semantics for the data. Ontologies use a formal mathematical model for defining semantics (e.g., description logic or first order logic). Hence, one can build a multi-graph, where different model elements are linked together using semantic edges defined by ontologies.

This is an important step towards both orienting data as well as harmonizing data in general. Without understanding the associated semantics of data, it is difficult (if not impossible) to ensure that the operation of the control loop will be correct.

[4.3.3.](#) Collaborate With Other Control Loops

Autonomic Systems MAY collaborate with other Autonomic Systems. This enables multiple Autonomic Systems to support each other, and work together to achieve goals that are mutually beneficial.

Strassner, et al.	Expires October 23, 2016	[Page 11]
-------------------	--------------------------	-----------

Internet-Draft	Autonomic Control Loop Usage	Apr 2016
----------------	------------------------------	----------

[5.](#) Control Loop Usage in Autonomic Networks

Autonomic systems use closed control loops. They may use one or multiple control loops to manage behavior; examples of these are the MAPE-K loop [[Kephart03](#)] and FOCAL [[Strassner07](#)] control loops, respectively.

Control loops operate to continuously observe and collect data that enables the autonomic management system to understand changes

to the behavior of the system being managed, and then provide actions to move the state of the system being managed toward a common goal. Self-adaptive systems move decision-making from static, pre-defined commands to dynamic processes computed at runtime.

Ideally, Autonomic Management will co-exist with traditional, or on-autonomic, management methods. This is because autonomic management will either be introduced in a greenfield environment (where it is the "only" management method), or more likely, in a hybrid environment that includes legacy systems and devices that are not capable of Autonomic Management.

[5.1.](#) Autonomic Management

In a hybrid environment, autonomic control loops are used to manage individual autonomic functions. In some hybrid environments (e.g., where a number of autonomic nodes are collaborating to provide a collective response to the system) and in many greenfield environments, autonomic control loops are used to manage not only functions, but processes and behaviors.

An autonomic control loop can be implemented using traditional and/or autonomic control mechanisms; examples include procedural and cognitive methods, respectively.

There are two types of behavior that are implied by the autonomic system coexisting with traditional systems: (1) autonomic methods can be used to manage legacy elements using traditional mechanisms (e.g., CLI, SNMP), and (2) autonomic methods can use a proxy to translate their management mechanisms into one or more forms that legacy elements can understand.

In principle, both of these approaches could be used by autonomic systems to manage autonomic elements. However, in practice, most autonomic systems will use autonomic mechanisms to manage autonomic elements, due to increased efficiency and expressivity.

Note that in either case, the basic control loop does NOT change. This is because the purpose of the control loop is to achieve its goals. Hence, it doesn't matter if new and/or legacy protocols are used, as long as the tasks can be accomplished.

[5.2.](#) Policy and Context

In FOCAL, Context is computed from information obtained and/or observed from the system being managed, along with other factors (e.g., business rules). This context information is used to determine the context that the system being managed is in. This context selects a working set of policies that are applicable for that context. The Policy Manager executes its policies, which define the behavior to be implemented by the Autonomic Manager. The Autonomic Manager then adjusts the set of control loop elements according to policy.

In the above simple example, if the current state equals the desired state, then no adjustment is necessary, so the control loop continues monitoring input data. In contrast, if the current states does not equal the desired state, then the control loop computes one or more plans, decides which plan(s) to execute, and then monitors the execution of the plan(s) to ensure that the expected outcomes occurred.

Machine learning algorithms monitor all of the operations of the control loop, building up a knowledge base that can correlate types of scenarios to solutions. It also records the efficacy of the remediations determined by the control loop processing.

[5.3](#) Types of Policies

The previous section showed the importance of using context-aware policies to control the processing of Autonomic control loops. There are two types of classifications of policies:

- 1) policies that pertain to specific actors, and
- 2) policies of a technological nature

[5.3.1](#). Policies Organized by Actors

The Policy Continuum [[Davy07](#)] defines a set of stratified policy languages, where each language is used by one or more actors in the end-to-end management of the system. This helps ensure consistency among the different constituencies that use policies, enabling each constituency to use a grammar and terminology that is familiar to them while being able to relate each language to at least each other language at the next lower (or higher) level of abstraction.

The purpose of the Policy Continuum is to emphasize that different actors think of policy differently. The essential point of the Policy Continuum is **not** the **number** of languages used, but rather the number of **actors** that **require** different concepts and terminology (and hence, different forms and structure of policy) to define the desired behavior of the system.

[5.3.2.](#) Policies Organized by Technology

The document [draft-strassner-supra-generic-policy-info-model-02](#) describes the difference between two types of policy rules. Imperative policies, typified by "condition-action" or "event-condition-action", define the set of commands to perform to manipulate the state of the system. In contrast, declarative policies, typified by logic-based languages, define relationships between variables in terms of functions or inference rules. A third type of policy rule, called a procedural policy, is one that explicitly defines a sequence of actions to execute given a set of conditions.

To date, the vast majority of policy implementations are either imperative or procedural. Lately, a lot of excitement has been generated over the concept of "intent-based" policies, which have been described as declarative policies.

[5.4.](#) Policy Conflicts

There are two classes of policy conflicts that must be taken into account if policy is to be used to control the processing of the control loop. They are:

- 1) Conflicts arising from technology, and
- 2) Conflicts arising from different actors

This is elaborated on in the following two subsections.

[5.4.1.](#) Policy Conflicts Caused by Technology

In imperative and procedural policies, policy conflict detection and remediation **MUST** be provided. Since state is directly manipulated by both types of these policies, different instances

of each can give rise to conflicting actions in response to the same conditions. For example, if two policies have the same conditions but different actions, this is a conflict.

There are many different algorithms to resolve policy conflicts. The simplest is adding a priority integer to each policy rule. This, however, is not advised, because:

- 1) it is complex to ensure that all integers are properly ordered for all cases, and
- 2) this is a static, reactive mechanism, and may not be able to be adjusted dynamically to resolve all conflicts

This type of policy conflict detection and resolution will be examined later in the lifecycle of the ANIMA WG.

Certain types of policy languages, such as logic-based declarative policies, do not need an explicit policy conflict detection process. This is because the logic itself ensures that policy conflicts are not allowed. A simple example is Datalog, which consists of a set of statements that determine whether a proposition is true or not.

[5.4.2.](#) Policy Conflicts Caused by Different Systems

Conflict can occur between the following broad classes of systems:

- o between actions of different autonomic networks
- o between actions of an autonomic network and actions of a non-autonomic network

[RFC7575] recommends the use of prioritization, which yields the following (incomplete) first pass of remediation:

- o manual, or operator-driven (e.g., using scripts) operations have the highest priority
- o operator-driven autonomic operations
- o default behavior of autonomic operations

< more in the next revision of this I-D >

[5.5.](#) Control Loops

Control loops provide a generic mechanism for self-adaptation. That is, as user needs, business goals, and the ANI itself change, self-adaptation enables the ANI to change the services and resources it makes available to adapt to these changes. Self-adaptive systems move decision-making from static, pre-defined commands to dynamic processes computed at runtime.

Control loops operate to continuously capture data that enables the understanding of the system, and then provide actions to move the state of the system toward a common goal.

[5.5.1.](#) Types of Control

There are two generic types of closed loop control. Feedback control adjusts the control loop based on measuring the output of the system being managed to generate an error signal (the deviation of the current state vs. its desired state). Action is then taken to reduce the deviation.

In contrast, feedforward control anticipates future effects on a controlled variable by measuring other variables whose values may be more timely, and adjusts the process based on those variables. In this approach, control is not error-based, but rather, based on knowledge.

Autonomic control loops MAY require both feedforward and feedback control, depending on the specific type of algorithm used.

[5.5.2.](#) Types of Control Loops

There are many different types of control loops. In autonomics, the most commonly cited loop is called Monitor-Analyze-Plan-Execute (with Knowledge), called MAPE-K [[Kephart03](#)]. However, MAPE-K has a number of systemic problems, as described in [[Strassner09](#)]. Thus, other autonomic architectures, such as AutoI [[AutoI](#)] and FOCALE [[Strassner07](#)] use different types of control loops. In these two cases, both AutoI and FOCALE evolved from the OODA control loop [[Boyd95](#)]. One of the most important reasons for using this loop, and not the MAPE-K loop, is because the OODA loop contains a

critical step not contained in other loops: orientation. Orientation determines how observations, decisions, and actions are performed. For example, assume that different types of sensor data need to be collected. Furthermore, assume that each type of sensor data uses a different data model. Orientation explicitly ensures that each set of sensor data is normalized to a common form. As another example, different data often have different semantics that affect their interpretation; orientation explicitly takes this into effect.

Figure 2 shows a simplified model of a control loop containing both feedforward and feedback elements.

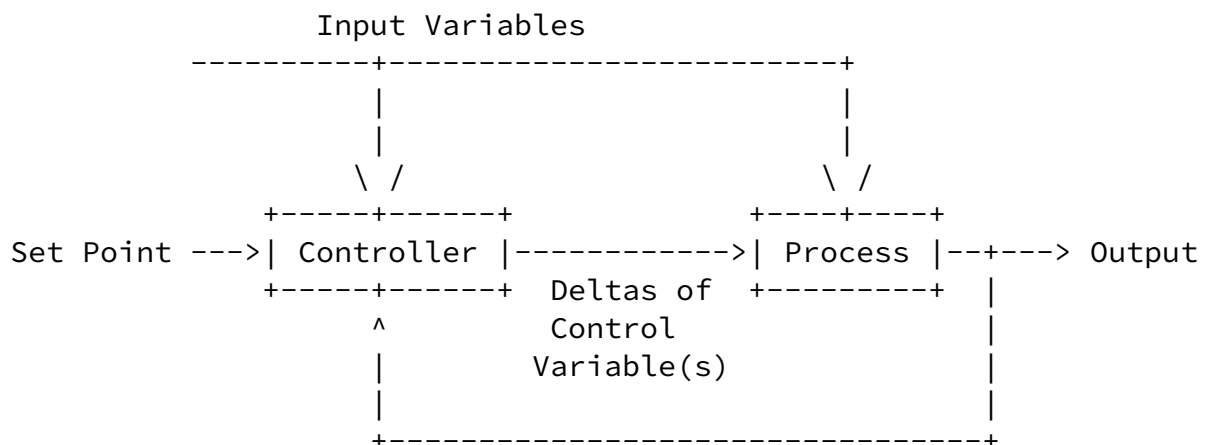
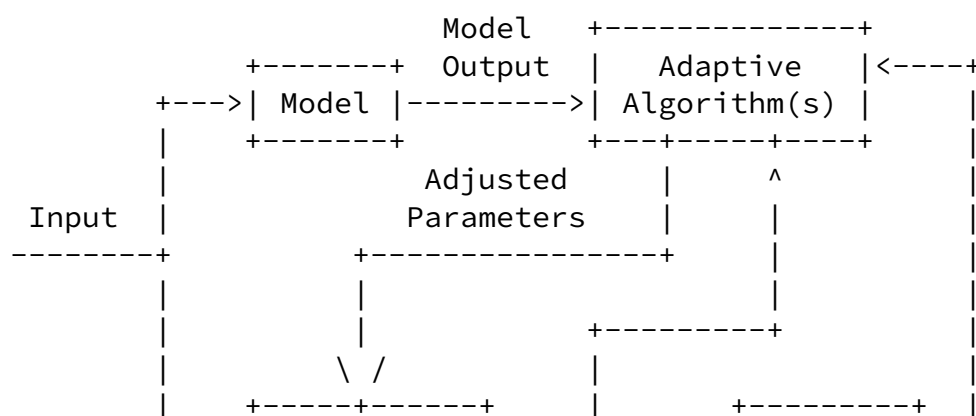


Figure 2: Control Loop with Feedforward and Feedback Elements

Note that Figure 2 is a STATIC model. Figure 3 is a dynamic version, called a Model-Reference Adaptive Control Loop (MRACL).



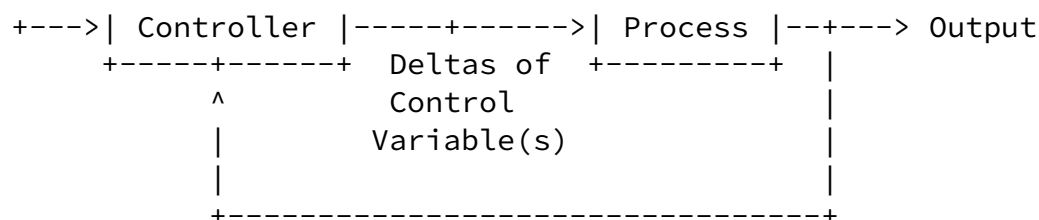


Figure 3: A Model-Reference Adaptive Control Loop

More complex adaptive control loops have been defined; these will be described in a future I-D, so that an appropriate gap analysis can be defined to recommend an architectural approach for ANIMA.

[5.5.3.](#) Management of an Autonomic Control Loop

Both standard and adaptive control loops (e.g., as represented in Figures 2 and 3, respectively) enable intervention by a human administrator or central control systems, if required. Interaction mechanisms include changing the behaviour of one or more elements in the control loop, as well as providing mechanisms to bypass parts of the control loop (e.g., skip the "decide" phase and go directly to the "action" phase of an OODA loop, as is done in FOCAL). This also enables the default behaviour to be changed if necessary.

[6.](#) Security Considerations

To be done in the next revision

[7.](#) IANA Considerations

This document requests no action by IANA.

[8.](#) Acknowledgements

TBD

[9.](#) References

[9.1.](#) Informative References

[[draft-ietf-anima-reference-model-01](#)]

Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Liu, B., Nobre, J., Strassner, J., "A Reference Model for Autonomic Networking", June 2015

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.

[RFC7575]

Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", [RFC 7575](#), June 2015.

[AutoI]

Galis, A., Denazis, S., Bassi, A., Giacomini, P., Berl, A., Fischer, A., de Meer, H., Strassner, J., Davy, S., Macedo, D., Pujolle, G., Loyola, J.R., Serrat, J., Lefevre, L., Cheniour, A., "Management Architecture and Systems for Future Internet Networks," in FIA Book: "Towards the Future Internet - A European Research Perspective". IOS Press, May 2009, pp. 112-122, ISBN 978-1-60750-007-0

[Boyd95]

Boyd, J.R., "The Essence of Winning and Losing", 28 June, 1995

[Davy07]

Davy, S., Jennings, B., Strassner, J., "The Policy Continuum - A Formal Model", Proc. of the 2nd Intl. IEEE Workshop on Modeling Autonomic Communication Environments (MACE), Multicon Lecture Notes, No. 6, Multicon, Berlin, 2007, pages 65-78

[Kephart03]

Kephart, J. and D. Chess, "The Vision of Autonomic Computing", IEEE Computer, vol. 36, no. 1, pp. 41-50, DOI 10.1109/MC.2003.1160055, January 2003.

[Mitchell97]

Mitchell, T., "Machine Learning", McGraw-Hill, March, 1997
ISBN 978-0070428072

[Strassner07]

Strassner, J., Agoulmine, N., Lehtihet, E., "FOCALE - A Novel Autonomic Networking Architecture", International Transactions on Systems, Science, and Applications (ITSSA) Journal, Vol. 3, No 1, pp 64-79, May, 2007

[Strassner09]

Strassner, J., Kim, S., Hong, J., "The Design of an Autonomic Communication Element to Manage Future Internet Services", Proc. of the 12th Asia-Pacific Network Operations and Management Conference, pg 122-132

Authors' Addresses

John Strassner
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA
Email: john.sc.strassner@huawei.com

Joel Halpern
Ericsson
P. O. Box 6049
Leesburg, VA 20178
Email: joel.halpern@ericsson.com

Michael H. Behringer
Cisco Systems
Building D, 45 Allee des Ormes
Mougins 06250
France
Email: mbehring@cisco.com

