A Systematic Survey on the Design of Self-Adaptive Software Systems Using Control Engineering Approaches

Tharindu Patikirikorala, Alan Colman, and Jun Han
Swinburne University of Technology
Victoria, Australia
{*tpatikirikorala,acolman,jhan*}*@swin.edu.au*

Liuping Wang
Royal Melbourne Institute of Technology
Victoria, Australia
liuping.wang@rmit.edu.au

Abstract—**Control engineering approaches have been identified as a promising tool to integrate self-adaptive capabilities into software systems. Introduction of the feedback loop and controller into the management system potentially enables the software systems to achieve the runtime performance objectives and maintain the integrity of the system when they are operating in unpredictable and dynamic environments. There is a large body of literature that has proposed control engineering solutions for different application domains, handling different performance variables and control objectives. However, the relevant literature is scattered over different conference proceedings, journals and research communities. Consequently, conducting a survey to analyze and classify the existing literature is a useful, yet a challenging task. This paper presents the results of a systematic survey that includes classification and analysis of 161 papers in the existing literature. In order to capture the characteristics of the control solutions proposed in these papers we introduce a taxonomy as a basis for classification of all articles. Finally, survey results are presented, including quantitative, cross and trend analysis.**

I. Introduction

In [16] Shaw compares the suitability of software engineering methodologies with control engineering methodologies to design a cruise control system and states: "…When the execution of a software system is affected by external disturbances-forces or events that are not directly visible to or controllable by the software—this is an indication that a control paradigm should be considered for the software architecture …". Many state-of-the-art software systems have become complex and large scale, and have to deal with unpredictable environmental conditions and dynamics. As a consequence, subsequent to the Shaw's paper, many papers (e.g., [3], [6], [9], [15]) have identified control engineering methodologies as a promising solution to implement self-adaptive software systems. The integration of the feedback loop and controller potentially enables operational goals to be achieved and costs related to human supervision to be reduced, while reacting to unpredictable disturbances and un-modeled system dynamics in a timely and effective manner.

The implementation of a control engineering solution consists of two major steps: (1) modeling the dynamics of the system and (2) developing a control system [10]. Many such control solutions have been proposed for software systems. However, these efforts are scattered over different conference proceedings, journals and research communities. They also relate to different application domains, and deal with different performance variables and control objectives. Analyzing the characteristics and trends of a large body of literature is a challenging task, consequently results of the exiting surveys [2], [4], [6]–[8], [11], [15], [20] are often significantly limited to application domains or solution domains.

The main objectives of this systematic survey are to (1) build a classification model of the existing literature, (2) quantify the published research work on the various modeling, control schemes and validation techniques utilized, and (3) analyze the clustering of papers across categories of the classification model and identify any apparent trends. Such analysis could aid future researchers and developers of self-adaptive systems to better identify the scope and domain applicability of various control engineering techniques and use proven modeling and control schemes for the control problem at hand. To achieve these objectives 161 papers were selected from different conference proceedings and journals. A taxonomy was subsequently developed in order to capture the content of these papers at a high-level of abstraction and then classify the literature in a systematic way. Finally, the results of the survey are presented with a quantitative, cross and trend analysis.

II. Related Work

There are several surveys related to this work that provide overviews of control engineering applications to manage software systems from different perspectives. An early survey [2] covers applications of feedback control in web servers, network, scheduling and storage management. However, this survey does not include works published after year 2003 in this area. In [7], a comprehensive survey has been conducted on different types of control engineering approaches applied to middleware (e.g., web and application servers). A limited set of key research works that used different types of control system designs to manage performance of software systems is presented in [20]. In addition, [6] provides a classification of a limited set of papers according to the performance attribute regulated by the control system. Brun et al in [3] also gives a classification based on the non-adaptive and adaptive control system designs for software systems. Furthermore, work in [14] classifies the literature according to the control system design techniques used. The detailed summary of several control engineering solutions proposed by several researchers can be found in [1], [10], [21]. Moreover, the lists of challenges and design rules when applying control engineering methodologies to software systems are presented in [8], [9].

The surveys [4], [15] analyze the attempts based on the software architectural approaches to implement the self-adaptive systems, giving less emphasis to the control engineering approaches.

In contrast to the above work, this systematic survey provides a comprehensive classification of the literature based on a taxonomy which includes properties of the target system, control system and validation mechanism.

III. Review Method

Kitchenham et al. in [12] provide a set of guidelines to conduct a systematic literature review. These guidelines include the steps for formulating research questions to be answered by the review and developing a review protocol. These guidelines are followed for instance in [5], [18], to conduct systematic reviews in different research areas of software engineering. In this work, we also follow the guidelines in [12] to conduct the systematic survey. The steps we followed in this survey are as follows:

A. Research Questions

Formulation of the research questions is the most important step of a systematic review [12]. The research questions addressed by this survey are:
RQ1: How can we classify the existing approaches based on the characteristics of the target software system (problem domain) and control system implemented (solution domain)?
RQ2: What are the methods used to model the dynamics of the software system?
RQ3: What are the control schemes, control system architectures and controllers (algorithms) used by the existing work?
RQ4: What are the techniques used to validate the proposed control solutions?
RQ5: What are the common clustering patterns and trends that exist in these proposed control solutions?

B. Review Protocol

Developing a review protocol is important, in order to select, organize and analyze the existing work without the possibility of bias. The review protocol is a planned set of activities [12], which includes the following steps.

1) Search Process: The basic idea behind this step is to decide on search strings and sources to search for the relevant papers (so called primary studies) for the survey. Deciding search strings for this survey was challenging because there is a large body of literature spanning the areas of software and control engineering. In order to maintain the count of the search results manageable, 'feedback control', 'QoS', 'software system' and 'software application' were used as the search strings and well known literature search engines like IEEE explore, ACM Digital library, ScienceDirect and DBLP Computer Science Bibliography were used to assist and narrow down the search process. In addition, based on our previous experience we also selected the conference proceedings and journals listed in Table I as sources. Furthermore, we also included all papers that cited the text book [10] (a total of 424) and the papers cited in surveys ([3], [6]–[8], [15]), related to feedback control or self-management systems. The papers gathered from this process were further investigated to improve the coverage by including the papers cited in the selected paper and other papers that cited the selected paper. Primarily the title, keywords and abstract of the paper were used to make a decision on the relevance of the paper. However, when it was inadequate to make a decision, the introduction and approach sections of the paper were considered as well.

2) Inclusion and Exclusion Criteria and Quality Assessment: The selected papers from the search process are further evaluated in this step to determine whether the selected studies are relevant in answering the research questions and to meet the expectations of the study.
Inclusion criteria: The main inclusion criteria were, (1) the date of publication—between 1^{st} of January 2000 to 1^{st} of November 2011, (2) language written—English and (3) problem domain and solution domain—papers that addressed the problems on automating the management of the software applications, middleware or environments that deployed software components (e.g. data centers), and those that propose solutions to these problems based on control engineering methodologies. In order to be a control engineering solution, we investigated the two major steps of control system design, i.e., modeling of system dynamics and controller implementation. The list of control engineering methodologies selected are covered in details in Section IV-B4. In addition, there were several papers that duplicated the same contributions in different papers or cases where the conference papers were extended to journal articles. Removing such duplications was a major challenge to avoid the bias of the systematic survey. In such cases, we included the most complete paper (e.g., journal version was included as oppose to the conference paper).
Exclusion criteria: There is a large body of literature which has used control engineering methodologies to automate

Table I: Conference proceedings and journals selected to gather papers

Source	Acronym
International Conference on Autonomic Computing	ICAC
International Symposium on Software Engineering for Adaptive and Self-Managing Systems	SEAMS
International Conference on Parallel and Distributed Systems	ICPADS
International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks	FeBID
International Workshop on Quality of Service	IWQoS
International Conference on High Performance Computing	HiPC
International Conference on High Performance Computing and Communications	HPCC
IEEE Real-Time and Embedded Technology and Applications Symposium	RTAS
American Control Conference	ACC
IEEE Conference on Decision and Control	CDC
IEEE Transactions on Parallel and Distributed Systems	PDS
IEEE Transactions on Network and Service Management	NSM

management of mobile, wireless and routing networks. These studies are out of the scope of this survey. The papers that proposed management systems without utilizing control theoretic approaches (e.g., optimization solution) were excluded as well. This also excludes the control solutions primarily based on fuzzy logic, neural networks, case based-reasoning and reinforcement learning. In addition, the papers that only deal with hardware (e.g, processor chips) or operating system level management issues with control solutions were also excluded. Furthermore, many papers that present the challenges, design guidelines and short surveys (e.g., [8], [21]) were excluded because they do not meet the major inclusion criteria.

Quality assessment: Assessing the quality of the paper or its contributions is a challenging and complex task. In order to evaluate the quality of the selected papers we used the following criteria

QA1: Is the paper peer-reviewed?

QA2: Does the paper provide a validation for the proposed solution?

If the answers to both these questions are '*yes*', we included the paper in this survey.

At the end of this step, 161 papers that met the above criteria were selected as the primary studies of this survey. The complete list of these papers can be found in [13].

3) Data Extraction: The next step is to finalize the data extraction strategies. In order to answer the research questions formulated in Section III-A, information has to be extracted from the selected papers accurately without any bias. The extracted data provides an abstract view and knowledge about a specific paper. To extract the data in a systematic and standardized way, we started off with a basic taxonomy, which was further developed during the data extraction process. The basic taxonomy was developed by the authors from their previous experience. The details of the final taxonomy are presented in Section IV.

Firstly, a *data extraction form* was documented based on the taxonomy (see Figure 1). This document also included other details of the paper such as the title, authors, conference/journal, publication year, bibliography identifier and additional notes. Then, each paper was read in detail and the data extraction form was filled. In addition, extracted data was rechecked to improve the accuracy of the data extraction. After the data extraction forms of all the papers were completed, a relational database schema was designed to record the data in a database management system. This is done to improve the accuracy and tractability of the classification and meta-analysis tasks involved in the next steps of the survey by using standard feature rich query languages provided by the database management systems. For this purpose we used the Microsoft SQL server. Next, all the information in the data extraction forms was inserted to the database.

4) Synthesis of the Extracted Data: The final step was to analyze the recoded data and answer the research questions and present the results of the survey. The details of the outcomes of this step is presented in Section IV and V.

IV. Taxonomy

This section presents the taxonomy we developed after the detailed analysis of the literature. This taxonomy provides a mechanism to extract the knowledge about a particular paper and represent it at a high-level of abstraction. It is also a tool to classify and mine clustering patterns that exist in the different levels of the hierarchy. The finalized taxonomy is shown in Figure 1. The first level of the taxonomy captures the characteristics of the *Target system*, *Control System* and *Validation* provided in the paper. These categories also have different subcategories. The final hierarchy of the taxonomy was developed by further refining the classifications during the data extraction in order to keep the taxonomy in a manageable size. The details of these categories are given below.

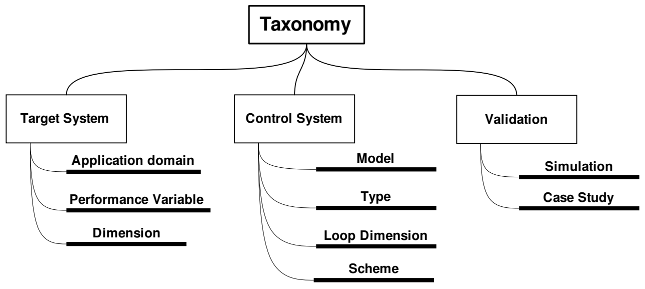


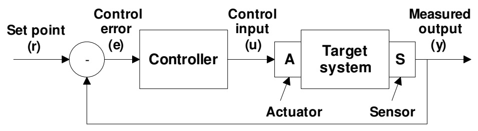
Figure 1: The high-level structure of the taxonomy

A. Target System

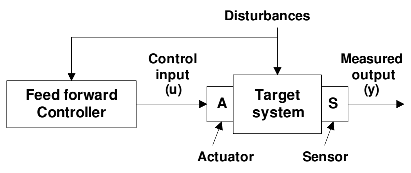
This category represents the characteristics of the software system controlled by the proposed control engineering solution in each primary study. It was further classified by the subcategories of *Application domain*, *Performance/controlled variables* and *Dimension* of the target system. The application domains extracted from the selected papers include data center, virtual machine environments, data storage, middleware and real-time systems. The control engineering solutions are primarily used to maintain the performance attributes at desired levels. Consequently, the performance/controlled variables of the target software system are the major property that has to be investigated. The performance/controlled variables were further classified in to Response time, Throughput, Progress/Miss ratio, Power utilization, Processor utilization, Hit rate/ratio, Memory utilization, Server utilization, Queue length, Temperature, Number in system, Scheduling error and Other. The dimension of the target system relates to the control objectives of the problem at hand. It can be classified as single-input-single-output (SISO) or multi-input-multi-output (MIMO), which achieves a single control objective or multiple control objectives respectively.

B. Control System

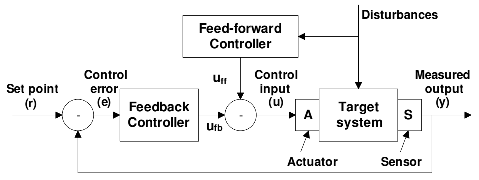
This category characterizes the control engineering solution proposed. The design and implementation of the control solution includes two basic steps. Firstly, the behavior of the target system has to be modeled. Secondly, a suitable control system has to be implemented. The details of design



(a) Feedback control system



(b) Feed-forward control system



(c) Feedback and Feed-forward control system

Figure 2: Block diagrams of different types of control systems

and implementation process are captured in the following subcategories.

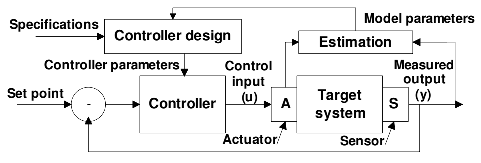
1) Model: The dynamics of a target system can be formally represented by the analytical (first-principle) or black-box models. The analytical models represent the behavior of the system by using the underlying physical laws governing the target system (for instance, mass-balance, electrical, friction laws). However, in the case of software systems, such models are not available or significantly complex [10]. From this survey, we observed the use of queuing models, which can be also classified as an analytical model to represent the behavior of many systems. As a consequence, queuing model was included as a subcategory. In contrast, the black-box models describe the system behavior with the input and output variables considering the system as a black-box. System identification (SID) is a widely used method to construct black-box models for a system. A SID experiment is conducted offline by applying a specially designed input signal on the system and to gather output data for a sufficient period of time [10]. Then the gathered measurements of the input and output data is used to estimate the model.

2) Type: Three types of control systems are as follows.

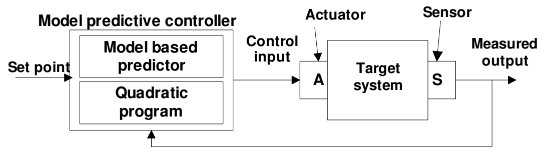
Feedback control system: Figure 2a shows a block diagram of a feedback control system. The target system provides a set of performance variables referred to as *measured outputs*. The sensor monitors the outputs of the target system, while the *control inputs* are sent to the actuator to change the behavior of the system by adjusting the system inputs. The feedback controller is the decision making unit of the control system. The main objective of the controller is to maintain the outputs of the system sufficiently close to the desired values, by adjusting the control inputs under disturbances. These desired values are translated in control system terms as the *set point signals*, which gives the option for the control system designer to specify the goals or values of the outputs that have to be maintained at runtime. The feedback control system is a reactive decision making mechanism, because it waits until a disturbance affects the outputs (feedback) of the system to make the necessary decisions.

Feed-forward control system: In contrast to feedback control, feed-forward control (See Figure 2b) measures the major disturbances and adjusts the inputs to reduce the impact of the disturbance on the system outputs. Consequently, it is considered as a proactive control mechanism in the literature. However, if the disturbance cannot be modeled accurately the performance of the feed-forward controller may be significantly poor. Furthermore, typically in the cases where all the disturbances cannot be measured or there are significant model uncertainties, the control objectives of maintaining the outputs around the set points (so called set point tracking) may not be achieved.

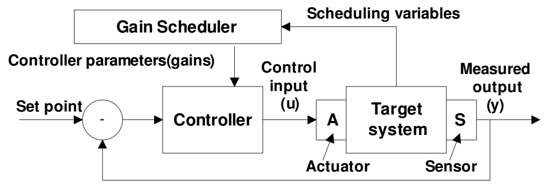
Feedback and Feed-forward control system: Figure 2c shows the architecture of a combined feedback and feed-



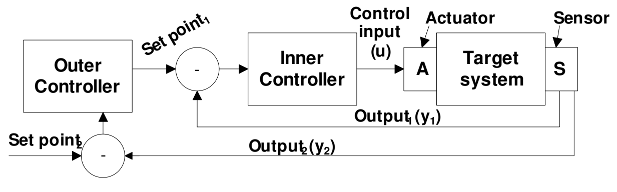
(a) Self-tuning adaptive control



(b) Model predictive control



(c) Gain scheduling control



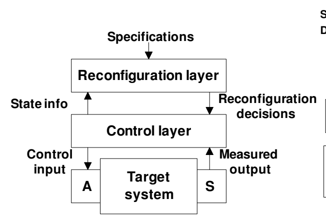
(d) Cascaded control

Figure 3: Block diagrams of basic schemes

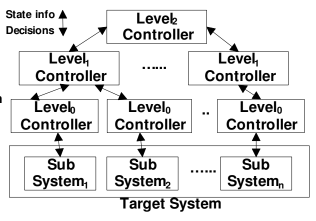
forward control system. It addresses the limitation of both schemes, where the feed-forward control adjusts the inputs based on the disturbances that are measurable, while the feedback control implements the set point tracking under unmeasured disturbances.

3) Loop Dimension: The design of a control system depends on the control objectives, i.e., whether it needs to achieve a single objective (SISO) or multiple objectives (MIMO). In the case of a SISO system a SISO control system is sufficient to achieve the objective. When there are multiple control objectives the control system that needs to be designed is complex. We observed mainly two solutions in our survey, including the design of multiple-SISO control systems/loops or a MIMO controller. A multiple-SISO control system decomposes the multiple control objectives into multiple single objectives and then designs multiple SISO control systems. In contrast, the MIMO control systems, achieves all the objectives using a single controller.

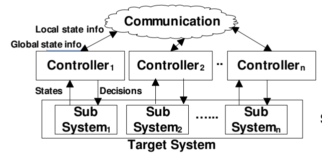
4) Scheme: This survey revealed that different control schemes have been used to implement the self-management capabilities into software systems. We further classify these schemes as basic and complex schemes. The difference is that the basic schemes have well defined structure and concrete implementation mechanisms, while complex schemes



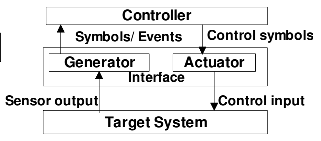
(a) Reconfiguring



(b) Hierarchical control



(c) Decentralized control



(d) Hybrid control

Figure 4: Conceptual architectures of the complex schemes

are conceptual control schemes realized in various methodologies.

a) Basic Schemes

Fixed-gain control: The structure of a fixed-gain control scheme is the same to that of Figure 2a. For instance, different variations of the Proportional Integral Derivative (PID) controller is one such fixed-gain controller widely used in existing work due to their robustness against modeling errors, disturbance rejection capabilities and simplicity [10]. The control algorithm of the PID controller is as follows: $u(k) = K_p e(k) + K_i \sum_{j=1}^{k} e(j) + K_d(e(k) - e(k-1))$, where $u(k)$ is the input for the current sample instance k, $e(k)$ is the difference between the output and set point and K_p, K_i and K_d are the parameters of the controller called *gains*. These gains are computed based on the model and other design specifications and remain fixed at runtime (consequently, the name, fixed-gain controller).

Adaptive control: In contrast to fixed gain control, adaptive control dynamically estimates the model parameters and gains of the controller at runtime. As shown in Figure 3a, adaptive controllers have a parameter adjustment loop, which derives these required parameters at runtime [17]. The parameters of the target system's model are estimated by the *Estimation* component, while the *Controller design* component uses these estimated model parameters and high-level control objectives provided by the designer to compute the gains of the controller.

Linear Quadratic Regulator (LQR): LQR is an optimal control strategy particularly useful in the MIMO control system designs. It uses a cost function, which represents a quadratic formula involving the control error and control effort. The basic idea is to minimize the cost function so that the error is minimized with a small control effort. It also gives the opportunity to trade-off between speed of response to disturbances and over-reacting to noisy output signals. For more details see [10].

Model predictive control (MPC): MPC is a class of control algorithms that perform on-line optimization with a natural ability to deal with the system constraints and its particularly useful to manage MIMO systems. It is similar to LQR, however the general idea behind MPC is to optimize the future behavior of the system outputs by computing the trajectory of the control inputs. Firstly, using the model of the system and the feedback signals, the behavior of the system outputs is predicted for a number of future samples. Then the objective of the MPC is to compute the trajectory of control inputs to maintain the predicted future outputs sufficiently close to the desired set point, subject to various constraints. For more details on MPC see [19]. MPC needs the system model and a standard quadratic programming solver to solve the optimization (or constraint) problem online as shown in Figure 3b.

Gain scheduling: Gain scheduling is also regarded as an adaptive control mechanism in [17]. Figure 3c shows the block diagram of a gain scheduling control system. In contrast to adaptive control, gain scheduling does not have a model estimation component. Instead, it uses a predefined logic/rule based evaluation to change the controller online. The predefined rules are implemented in the gain scheduling component depending on the prior knowledge about the performance variables, disturbances and conditions. At runtime when the rules are satisfied the relevant controller gains are updated in the controller by the gain scheduling component.

Cascaded (nested) control: Most of the approaches assume that the set point specified in the controller remains constant or changes infrequently. The main objective of a cascade control mechanism (Figure 3d) is to change the set point of the inner loop. The outer loop tries to maintain a single output around the set point by mapping control objective into the inner loop control problem. Depending on the control error of the outer loop, it generates the set point periodically for the inner loop. When the inner loop achieves its new set point, the control objectives of the outer loop will be indirectly achieved at the same time.

b) Complex Schemes

Reconfiguring control: In the adaptive control schemes the controller algorithm and the organization of the components in the loop stays fixed overtime [14]. However, for different operating conditions and disturbances different control algorithms or loop organizations may provide better control [14]. In reconfiguring control schemes the main idea is to change the control algorithms, models and architecture of the control system to deal with the changing operating regions of the target system. Figure 4a illustrates the layered architecture of reconfiguration control. The control layer consists of the control system (including the controller) providing the control in the current time instance. The responsibility of the reconfiguration layer is to reconfigure the architecture of the control layer (e.g., by changing controller) so that the control objectives of the target system can be achieved under requirement or environmental changes.

Hierarchical control: Figure 4b shows a general architecture of the hierarchical control scheme. The hierarchical control schemes can be used to realize control objectives of large distributed systems. The main idea is to implement the divide-and-conquer concept, where lower level (Level$_0$) controllers manage the sub systems of a large system, while high-level controllers act as a coordination layer of the lower level control systems. For instance, high-level controllers may adjust the control objectives of the lower level controllers after looking at the system-wide control objectives.

Decentralized control: In contrast to the hierarchical control where the management decisions flow downwards from a centralized management entity, the decentralized control manages each subsystem with a controller individually. There is no centralized entity that looks at the global control objectives and specifies the management objectives. Instead, the communication layer provides the information about the global state variables or just the states of the neighboring subsystems (see Figure 4c). Then, utilizing the local states and information from the communication layer, each individual controller provides control in an independent manner. As a consequence, the system-wide objectives are/may be achieved in a decentralized fashion.

Hybrid control: Many software systems shows combined event and time based dynamics. The idea behind hybrid control is to incorporate both event and time based dynamic aspects into the control system design. Figure 4d shows a basic architecture of a hybrid control system. The *generator* produces special events after analyzing data from the target system. The controller operates with a target system model, typically described by a finite automata (hybrid automata). It begins from the starting state and moves through different states depending on the events/symbols generated by the generator. Corresponding to the state, the controller sends the control decisions as control symbols to the *actuator*, which converts them to the system inputs.

Some papers included in the survey implement control solutions, which included multiple control schemes together. In such cases we classified the papers under all the relevant schemes.

C. Validation

This category represents the type of validation provided in the paper to show the effectiveness of the proposed control engineering solution. It was further classified into validation based on a *Simulation* or *Case study*. A simulation based validation relies on some kind of a simulation model of a target system and then implementing the proposed solution on it. To develop a simulation model well established techniques like discrete-event simulations or off-the-shelf simulation tools (e.g., MATLAB) can be utilized. Case study

Table II: Quantitative results of the subcategories of 'Target system' category

Application Domain		
	Number of papers	Percentage
Middleware	54	30.9
Real-time systems	38	21.7
Data center	35	20.0
VM	24	13.7
Data Storage	23	13.1
Other	1	0.6
Performance variable		
	Number of papers	Percentage
Response time	76	37.8
Processor Utilization	40	19.9
Power Utilization	20	10.0
Progress/Miss ratio	17	8.5
Throughput	13	6.5
Hit rate/ratio	7	3.5
Queue length	5	2.5
Memory	4	2.0
Server utilization	4	2.0
Temperature	3	1.5
Tardiness	2	1.0
Number in system	2	1.0
Scheduling error	2	1.0
Other	6	3.0
Dimension		
	Number of papers	Percentage
MIMO	95	59.0
SISO	66	41.0

based validations implement a target system close to the real world settings and deploying the system in a physical environment. Then, that system is used to validate the proposed control solution. We further analyzed the case-study subcategory by extracting information on what benchmark software applications and/or workload generators were used.

V. Survey Results

The main focus of this section is to answer the research questions **RQ2**, **RQ3**, **RQ4** and **RQ5** formulated in Section III-A. To serve this purpose, Sections V-A, V-B and V-C provide a quantitative analysis of each high-level category of the taxonomy. In addition, Sections V-D and V-E respectively present a cross analysis and observed trends. Furthermore, the limitations of this survey are also listed in Section V-F. Note that the grouping of the paper references according to the taxonomy is not presented due to space limitations (for more details see [13]).

A. Analysis of the 'Target system' Category

Application Domain: Table II indicates that many control theoretic solutions are proposed to manage middleware (e.g., web servers and application servers). Similarly, for some of the management problems involved with real-time systems and data centers, the control engineering solutions have been proposed in close to 20% of the papers. In the case of real-time systems, another interesting observation was that all the control solutions were proposed to manage soft-deadlines in unpredictable environments as oppose to hard-deadlines. The main reason for this observation is under unpredictable disturbances, the deadlines of some tasks could be violated, which is not tolerated in the hard real-time systems. Furthermore, over 10% of the papers have investigated the management issues of data storage (e.g., databases, memory and cache) and virtual machine environments.

Performance variables: Table II illustrates the performance variables of the target system controlled by the control solution proposed in the papers. 19 different performance variables were identified. The first 13 attributes have been used in more than 1 paper, while the *Other* category represents 6 different performance variables appearing in 6 different papers. From the statistics the response time is one of the major performance attributes investigated in the existing literature. The reasons for this could be that the response time is (1) the user perceived performance attribute of the system (2) one of the attribute specified in SLAs and (3) useful to formulate a set point tracking control problem. The processor and power utilization are the other performance variables looked at by a large number of papers. In contrast, all other performance attributes are utilized in less than 10% of the papers. It is also evident that many variables related to queuing models are also used as the performance variables (e.g., queue length, server utilization and number in system).

Dimension: The simple classification based on the target system input-output dimension indicates that most of the target software systems are MIMO systems (59% of the papers, compared to 41% classified under SISO systems, see Table II). This further shows that there are typically multiple control objectives in the management problem of a software system.

B. Analysis of the 'Control system' Category

Model: From the statistics the black-box models are more popular than the analytical models. In close to 65% of the papers, the black-box models have been utilized. In contrast, the queuing and other analytical models have been used in similar percentages among the papers. The black-box models may have been more useful to directly capture the dynamics of the software systems because the complexity of the analytical models limits the use of well-established control engineering methodologies to design a controller, unless linearization and other approximation techniques are used to simplify the analytical models.

Type: Apart from the two papers which used stand-alone feed-forward control path, close to 99% the papers have included a feedback control loop. Many papers that used feed-forward control path had measured the workload rates as the primary disturbance. This is in fact true in most cases, however it is hard to accurately measure the workload rates because of the stochastic nature of the workloads faced by the software systems. In addition, there are other un-modeled disturbances such as garbage collections, runtime compiler

Table III: Quantitative results of the subcategories of 'Control system' Category

Model		
	Number of papers	Percentage
Black box	107	64.8
Queuing	30	18.2
Analytical model	28	17.0
Type		
	Number of papers	Percentage
Feedback	142	88.2
Feedback + forward	17	10.6
Feed-forward	2	1.2
Loop Dimension		
	Number of papers	Percentage
SISO	74	45.4
MIMO	50	30.7
Multi-SISO	39	23.9
Scheme		
	Number of papers	Percentage
Fixed	66	29.7
Adaptive	33	14.9
LQR	25	11.3
MPC	24	10.8
Hierarchical	17	7.7
Gain scheduling	16	7.2
Cascade	14	6.3
Hybrid	11	5.0
Reconfiguring	10	4.5
Decentralized	6	2.7

optimizations and competition for resources between components that would affect the performance of the feed-forward control. As a consequence, the feedback loop has been used by most of the cases to achieve the desired control objectives (set points) under un-modeled dynamics.

Loop dimension: The dimension of the controller or the control loop also reveals interesting results (see Table III). Many control solutions proposed in literature so far deal with a single control objective. 74 papers in total designed SISO control solutions. However, 89 papers have looked at MIMO control problems and proposed multi-SISO or MIMO control solutions for them. As mentioned, the software systems typically require multiple performance objectives to be achieved, which cannot be effectively handled by a SISO control system prompting an interest in more complete MIMO control solutions.

Scheme: From Table III, it is evident that many papers (66 in total) have utilized fixed gain (PID control variations) in the control solutions. The reason for this may be the simplicity and robustness of this control scheme. From the variations of the PID control, PI (50%) and I (24%) control algorithms have been used widely compared to other variations. The adaptive control, MPC and LQR schemes are utilized in more than 10% of the papers. Although, the large scale software systems and typical operating conditions and disturbances faced by the software systems demand complex control solutions, the complex schemes such as hierarchical, cascade and reconfiguring control have not been widely adopted compared to the basic control schemes.

Table IV: Quantitative results of the subcategories of 'Validation' category

Validation Method	Number of papers	Percentage
Case study/test bed	116	72.2
Simulation	54	33.5

Table V: Quantitative results of the case studies that used a benchmark and workload generator

Benchmark	Number of papers
TPC-W (http://rubis.ow2.org/)	11
Rubis (http://rubis.ow2.org/)	9
Trade6 (http://www.ibm.com/developerworks/data/tutorials/dm0506lau/)	5
RUBBoS (http://jmob.ow2.org/rubbos.html)	4
Workload generator	**Number of papers**
httpref (http://www.hpl.hp.com/research/linux/httperf/)	19
SURGE (http://www.net.t-labs.tu-berlin.de/~joerg/nsweb/doku/node12.html)	15
Benchmark workload generator	13
SPEC (http://www.spec.org/benchmarks.html)	5
Apache Ab (http://httpd.apache.org/docs/2.0/programs/ab.html)	3
Apache Jmerter (http://jmeter.apache.org/)	1

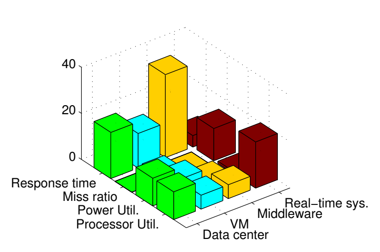
C. *Analysis of the 'Validation' Category*

The proposed control approach in each paper has been validated either by a simulation study or case study with a test bed (9 papers have used both) (see Table IV). The case study based validation method is the more widely adopted validation technique. In addition to the above statistics, we further analyzed the case studies that utilized or included the benchmark software systems in their test bed. It is worth noting that only a few papers either used or specified such usage of benchmarks. Table V summarizes the results of those papers. It lists the benchmarks that have been used in more than one paper. From the papers that used case studies, 27 papers have used a single or multiple benchmarks in their validations. The TPC-W benchmark has been used by 11 papers. Other benchmarks have been used in less than 10 papers.

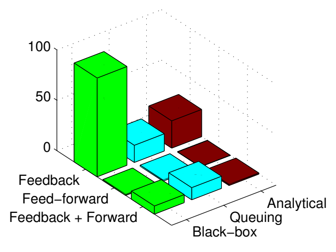
One of the important tools to provide validation is the workload generator. The statistics of the papers that mention such use are summarized in Table V. The workload generators such as *httpref* and *SURGE* that are not based on any benchmark have been used in 19 and 15 papers respectively. Furthermore, the workload generators can be classified as *open-loop* or *closed-loop*. For instance, *httpref* and *SURGE* are open-loop workload generators that send the requests without considering the completion of the previous requests sent by a particular user. In contrast, the workload generator provided by the *Rubis* benchmark simulates closed-loop workloads, i.e., the next request is sent based on the completion of the previous request of a particular user.

D. *Cross Analysis*

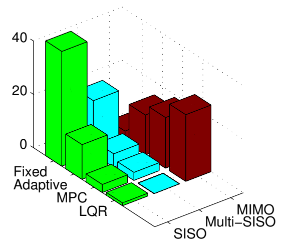
Figures 5a, 5b and 5c illustrate cross analysis of some of the subcategories of the taxonomy. The response time has



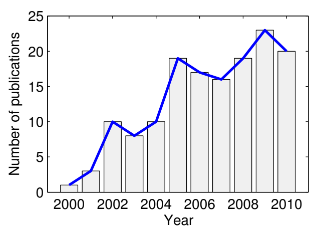
(a) Performance variable vs Domain



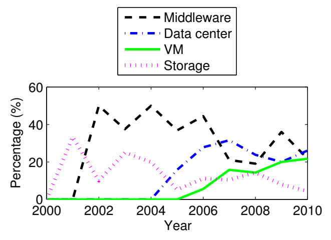
(b) Type vs Model



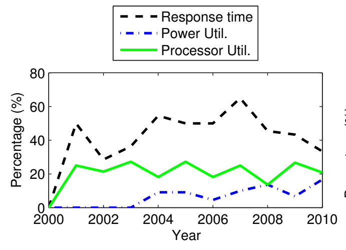
(c) Scheme vs Dimension



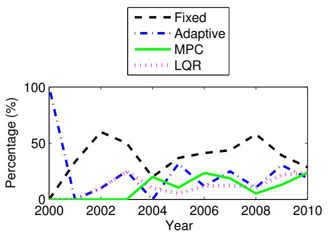
(d) Publications in each year



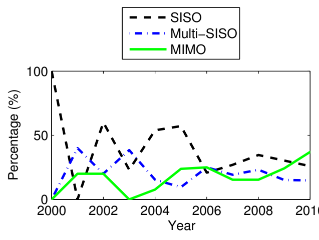
(e) Domain



(f) Performance variable



(g) Scheme



(h) Dimension

Figure 5: Cross and trend analysis of subcategories of the taxonomy

been used widely across all application domains, middleware being the highest (see Figure 5a). In addition, the power and processor utilization are used as the control variables in the domains of data center and VM environments. In contrast, the variables such as miss ratio and processor utilization have been used widely in the case of real-time systems. Such observations indicate that the managed performance attributes vary depending on the application domain and have to be carefully selected. There is also a correlation between how the system model is developed with respect to the type of the control system designed. According to Figure 5b, feedback control loops have been mostly designed based on a black-box model. Furthermore, when the feed-forward path is integrated to the control system, queuing models have been used to predict the behavior of the system. In Figure 5c, the control schemes are analyzed with respect to loop dimension. It is evident that the SISO and multi-SISO control systems have been widely developed using fixed gain (PID) control schemes. In contrast, MIMO control systems are mostly developed using MPC or LQR, indicating their ability to deal with MIMO control problems. Although it is not shown in Figure 5c, all the cascade control systems are designed as multi-SISO control systems, whereas the hierarchical and decentralized control systems are designed as multi-SISO or MIMO control systems. If the validation technique and application domain are compared, the case study based validations are popular across all the domains, apart from the real-time systems. In the case of real-time systems, the simulation studies have been widely used as well.

E. Trends

Figures 5e, 5f, 5g, and 5h show the percentages of work done in some of the subcategories from the total work across all subcategories in each year[1]. One of the main observations from Figure 5d is that the applications of the control engineering methodologies have increased in the last ten years. A contributing factor may be the increasing trend in investigating control engineering solutions for the problems in application domains such as data centers and virtual machine environments (See Figure 5e). Figure 5f illustrates increasing trends in performance variables such as response time and power utilization. The increasing demand for delivery of quality of service attributes such as response time, the inclusion of such requirements in the service level agreements and penalties involved in violations of these requirements may have affected this trend. Furthermore, the costs of power and demands for green computing may have triggered the investigation of control solutions to manage performance variables such as power utilization. There are increasing trends in the application of adaptive, LQR and MPC control schemes (see Figure 5g). It is hard to decide on the trends of other complex control schemes because of the lack of applications (data). Compared to SISO solutions, investigation of MIMO solutions have gathered momentum in the recent years according to Figure 5h. It is an interesting statistic that indicates a shortcoming in the efficiency of earlier attempts at SISO solutions. In the case of validation category there is no significant treads in both of the subcategories.

[1]The statistics of year 2011 were excluded because the publications of the entire year were not covered by this survey.

F. Limitations of the Survey

The search process of the papers was a manual process based on the title, key words and abstract. There may be cases where some papers were not included in the search results because of the lack of standards in such attributes of the paper. Consequently, this survey may not have exhaustively covered the entire literature. Hence, the quantitative results provided by this survey are really only valid based on the 161 papers included in the survey. Another challenging task was to remove the duplication of contributions across several papers. This is also a subjective process which may have introduced bias in some areas of the survey. Finally, as with all categorical schemes, the taxonomy developed is also, to some extent, subjective. To mitigate this limitation we started off with a basic structure and then extended the taxonomy during the data extraction process.

VI. Conclusions

In the existing literature many self-adaptive software systems have been implemented based on control engineering methodologies. This paper provides the details of a systematic survey of such control engineering approaches proposed in 161 papers in the literature. A classification model was built to capture and represent the information about each paper at a high-level of abstraction. The quantitative results were also presented and analyzed, providing an insight into the scope, extent and trends of research based on the categories of the classification model.

References

[1] T. Abdelzaher, Y. Diao, J. L. Hellerstein, C. Lu, and X. Zhu, “Introduction to control theory and its application to computing systems,” in *International Conference on Measurement and Modeling of Computer Systems*, 2008.

[2] T. Abdelzaher, J. Stankovic, C. Lu, R. Zhang, and Y. Lu, “Feedback performance control in software services,” *IEEE Control Systems*, vol. 23, no. 3, pp. 74 – 90, 2003.

[3] Y. Brun, G. Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, “Software engineering for self-adaptive systems.” Springer-Verlag, 2009, ch. Engineering Self-Adaptive Systems through Feedback Loops, pp. 48–70.

[4] B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, el al., “Software engineering for self-adaptive systems.” Springer-Verlag, 2009, ch. Software Engineering for Self-Adaptive Systems: A Research Roadmap, pp. 1–26.

[5] T. Dybå and T. Dingsøyr, “Empirical studies of agile software development: A systematic review,” *Inf. Softw. Technol.*, vol. 50, pp. 833–859, 2008.

[6] J. Guitart, J. Torres, and E. Ayguadé, “A survey on performance management for internet applications,” *Concurr. Comput. : Pract. Exper.*, vol. 22, pp. 68–106, 2010.

[7] R. Gullapalli, C. Muthusamy, and V. Babu, “Control systems application in Java based enterprise and cloud environments-a survey,” *International Journal of Advanced Computer Science and Applications*, vol. 2, pp. 103 –113, 2011.

[8] J. Hellerstein, S. Singhal, and Q. Wang, “Research challenges in control engineering of computing systems,” *IEEE Transactions on Network and Service Management*, vol. 6, no. 4, pp. 206 –211, december 2009.

[9] J. Hellerstein, “Challenges in control engineering of computing systems,” in *American Control Conference*, vol. 3, 30 2004-july 2 2004, pp. 1970 –1979 vol.3.

[10] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. John Wiley and Sons, 2004.

[11] M. C. Huebscher and J. A. McCann, “A survey of autonomic computing-degrees, models, and applications,” *ACM Comput. Surv.*, vol. 40, pp. 7:1–7:28, 2008.

[12] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.

[13] T. Patikirikorala, “A systematic survey on the design of self-adaptive software systems using control engineering approaches,” Tech. Rep., 2012, http://www.ict.swin.edu.au/personal/tpatikirikorala/Research.htm.

[14] T. Patikirikorala, A. Colman, J. Han, and L. Wang, “A multimodel framework to implement self-managing control systems for QoS management,” in *International symposium on Software engineering for adaptive and self-managing systems*, 2011, pp. 218–227.

[15] M. Salehie and L. Tahvildari, “Self-adaptive software: Landscape and research challenges,” *ACM Trans. Auton. Adapt. Syst.*, vol. 4, pp. 14:1–14:42, May 2009.

[16] M. Shaw, “Beyond objects: a software design paradigm based on process control,” *SIGSOFT Softw. Eng. Notes*, vol. 20, pp. 27–38, January 1995.

[17] K. J. strom and B. Wittenmark., *Adaptive Control*. Addison-Wesley Publishing Company, 1995.

[18] M. Turner, B. Kitchenham, P. Brereton, S. Charters, and D. Budgen, “Does the technology acceptance model predict actual use? a systematic literature review,” *Information and Software Technology*, vol. 52, pp. 463 – 479, 2010.

[19] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*. Springer Publishing Company, Incorporated, 2009.

[20] C. A. Yfoulis and A. Gounaris, “Honoring SLAs on cloud computing services: a control perspective,” in *European Control Conference*, ser. ECC09, 2009.

[21] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin, “What does control theory bring to systems research?” *SIGOPS Oper. Syst. Rev.*, vol. 43, pp. 62–69, January 2009.