UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

PLAN DE TRABAJO DE GRADO

FECHA DE PRESENTACIÓN: Bucaramanga, 1 de enero de 2023

TÍTULO: Mecanismos de adaptación autonómica de arquitectura software para la

plataforma Smart Campus UIS

MODALIDAD: Trabajo de investigación

AUTOR: Daniel David Delgado Cervantes, 2182066

DIRECTOR: PhD. Gabriel Rodrigo Pedraza Ferreira, Escuela De Ingeniería De

Sistemas e Informática

CODIRECTOR: MSc. Henry Andrés Jiménez Herrera, Escuela De Ingeniería De

Sistemas e Informática

ENTIDAD INTERESADA: Universidad Industrial de Santander

TABLA DE CONTENIDO

1

2

3

INTRODUCCIÓN

2

2

PLANTEAMIENTO Y JUSTIFICACIÓN DEL PROBLEMA

OBJETIVOS

3

3

3

3

3

.1 OBJETIVO GENERAL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

.2 OBJETIVOS ESPECÍFICOS . . . . . . . . . . . . . . . . . . . . . . . . . . .

4

MARCO DE REFERENCIA

4

4

5

6

6

6

6

6

6

6

6

7

4

.1 COMPUTACIÓN AUTONÓMICA . . . . . . . . . . . . . . . . . . . . . . . .

4

4

4

.1.1 MAPE-K . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

.1.2 MECANÍSMOS DE DESCRIPCIÓN . . . . . . . . . . . . . . . . . . .

.1.3 MECANÍSMOS DE ADAPTACIÓN . . . . . . . . . . . . . . . . . . .

4

4

4

.2 SISTEMAS EMBEBIDOS . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4

4

.2.1 INTERNET OF THINGS . . . . . . . . . . . . . . . . . . . . . . . . .

.2.2 SMART CAMPUS . . . . . . . . . . . . . . . . . . . . . . . . . . . .

.3 NOTACIÓN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4

4

.3.1 GRAMÁTICA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

.3.2 SERIALIZACIÓN DE DATOS . . . . . . . . . . . . . . . . . . . . . .

.4 ALGORITMIA DE COMPARACIÓN DE GRAFOS . . . . . . . . . . . . . . .

5

METODOLOGÍA

8

8

9

9

5

5

5

5

.1 AMBIENTACIÓN CONCEPTUAL Y TECNOLÓGICA . . . . . . . . . . . . .

.2 DEFINICIÓN DE LA NOTACIÓN DE LA ARQUITECTURA . . . . . . . . . .

.3 MECANISMOS DE COMPARACIÓN . . . . . . . . . . . . . . . . . . . . . .

.4 MECANISMOS DE ADAPTACIÓN . . . . . . . . . . . . . . . . . . . . . . . 10

.4.1 ACTIVIDADES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

.5 VALIDACIÓN DE RESULTADOS . . . . . . . . . . . . . . . . . . . . . . . . 10

5

5

6

7

8

CRONOGRAMA

PRESUPUESTO

BIBLIOGRAFÍA

11

11

12

1

MECANISMOS DE ADAPTACIÓN AUTONÓMICA DE ARQUITECTURA SOFTWARE

PARA LA PLATAFORMA SMART CAMPUS UIS

1

INTRODUCCIÓN

Dentro de la computación distribuida, una de las tendencias recientes dentro de la

industria, es la búsqueda de maneras de reducir la complejidad de administrar los

sistemas computacionales. A medida que estos crecen, en términos de tamaño y

extensión, el costo humano de la administración de los sistemas se vuelve insostenible.

En respuesta a esto, diferentes enfoques han surgido. Uno de estos es el autonomic

computing (computación autonómica). Planteada originalmente por IBM en el año 2001,

se presentaba como una posible solución a la problemática a partir del diseño y la

implementación de componentes auto-gestionados (Kephart, 2011).

Este acercamiento, aunque inicialmente concebido únicamente para sistemas

distribuidos, puede también ser particularmente útil en otras ramas como lo son las

2

PLANTEAMIENTO Y JUSTIFICACIÓN DEL PROBLEMA

La complejidad de los sistemas software ha ido en aumento (Horn, 2001, pp. 4-5). A

medida que se hace la transición a arquitecturas orientadas a microservicios (Forrester

Research, 2019); la computación distribuida es más común gracias a las soluciones

cloud (Loukides, 2021) y la computación embebida se hace más presente(Deichmann,

Doll, Klein, Mühlreiter, y Stein, 2022); la administración y gestión de estos requiere de

una mayor cantidad de recursos en términos técnicos y humanos con el ﬁn de

mantenerlos en los estados más óptimos respecto a los requerimientos del negocio. La

búsqueda de reducir o abstraer la complejidad de la gerencia de estos sistemas se ha

convertido en una necesidad (Lalanda, Diaconescu, y McCann, 2014).

Esta necesidad, así mismo, se presenta en los campos del Internet de las Cosas (IoT).

Es en esta área de la computación embebida donde, debido a las cambiantes

condiciones del mundo real, la arquitectura de estos sistemas de software se ve

constantemente afectada. Una de las posibles soluciones se encuentra en la

computación autonómica. Desde este enfoque, se tiene como objetivo sistemas con la

capacidad de auto-gestión, es decir, sistemas con la capacidad de manejarse a ellos

mismos dependiendo de las necesidades y las metas establecidas por los

administradores del sistema (McCann y Huebscher, 2004).

Ahora, tenemos el caso de Smart Campus UIS, una plataforma de IoT de la Universidad

Industrial de Santander. Esta ha realizado implementaciones parciales de una

arquitectura autonómica con capacidad de auto-describirse (Jiménez, Cárcamo, y

Pedraza, 2020). Dicho esto, y en búsqueda dar continuidad con los esfuerzos de

2

desarrollo realizados en la plataforma, el siguiente paso a dar está en la implementación

de mecanismos de adaptación los cuales le concedan las propiedades de

auto-conﬁguración y auto-sanación.

3

3

OBJETIVOS

.1 OBJETIVO GENERAL

Implementar un conjunto de mecanismos autonómicos para permitir la adaptación

de la Arquitectura Software IoT respecto a un modelo objetivo en la plataforma

Smart Campus UIS

3

.2 OBJETIVOS ESPECÍFICOS

Proponer una notación (lenguaje) para describir una arquitectura objetivo de un

sistema software I oT.

Diseñar un mecanismo para determinar las diferencias existentes entre una

arquitectura actual en ejecución y una arquitectura objetivo especiﬁcada.

Diseñar un conjunto de mecanismos de adaptación que permitan disminuir las

diferencias entre la arquitectura actual y la arquitectura objetivo.

Evaluar la implementación realizada a partir de un conjunto de pruebas con el ﬁn

de establecer la efectividad de los mecanismos usados.

3



4

MARCO DE REFERENCIA

Como base para el desarrollo del proyecto se encuentra en el establecer los

fundamentos necesarios para realizar la implementación de los mecanismos de

adaptación para seleccionar. Siendo así, es necesario conocer los principios de la

computación autonómica, las aplicaciones de la misma en la industria y las partes

requeridas para la integración.

4

.1 COMPUTACIÓN AUTONÓMICA

El concepto de computación autonómica, deﬁnido inicialmente por IBM (2001), se reﬁere

a un conjunto de características que presenta un sistema computacional el cual le

permite actuar de manera autónoma o auto-gobernarse, con el ﬁn de alcanzar algún

objetivo establecido por los administradores del sistema.

Los 8 elementos clave, deﬁnidos por IBM, que presentan este tipo de sistemas

son:

1

2

3

4

. Auto-conocimiento: habilidad de

conocer su estado actual, las

interacciones del sistema.

5. Auto-protección: facultad de

protegerse a sí mismo de ataques

externos.

. Auto-conﬁguración: capacidad de

reconﬁgurarse frente a los constantes

cambios en el entorno.

6. Auto-conciencia: posibilidad de

conocer el ambiente en el que el

sistema se encuentra.

. Auto-optimización: búsqueda

constante de optimizar el

funcionamiento de sí mismo.

7. Heterogeneidad: capacidad de

interactuar con otros sistemas de

manera cooperativa.

. Auto-sanación: aptitud de restaurar el

sistema en el caso de que se

presenten fallas.

8. Abstracción: ocultar la complejidad a

los administradores del sistema con

objetivos de alto nivel de abstracción.

Considerando lo anterior, se busca reducir la complejidad de la administración de los sistemas computacionales generada por el crecimiento de estos, como también la cantidad de recursos técnicos y humanos requeridos para su funcionamiento.

..

4

4

.1.1 MAPE-K

IBM, en cuanto a la implementación de las características, propone un modelo de ciclo

auto-adaptativo, denominado MAPE-K . En la Figura 1 se presentan las etapas que el

manejador debe seguir para así manejar cada uno de los elementos del sistema

computacional basado en una base de conocimiento común (Krikava, 2013).

Figura 1: El ciclo auto-adaptativo MAPE-K propuesto por IBM.

(Gorla, Pezzè, Wuttke, Mariani, y Pastore, 2010)

Cada una de estas fases son:

Monitorear (M): Esta fase se compone de la recolección, ﬁltración y reportar la

información adquirida sobre el estado del elemento a manejar.

Analizar (A): La fase de análisis se encarga del interpretar el entorno en el cual se

encuentra, el predecir posibles situaciones comunes y diagnosticar el estado del

sistema.

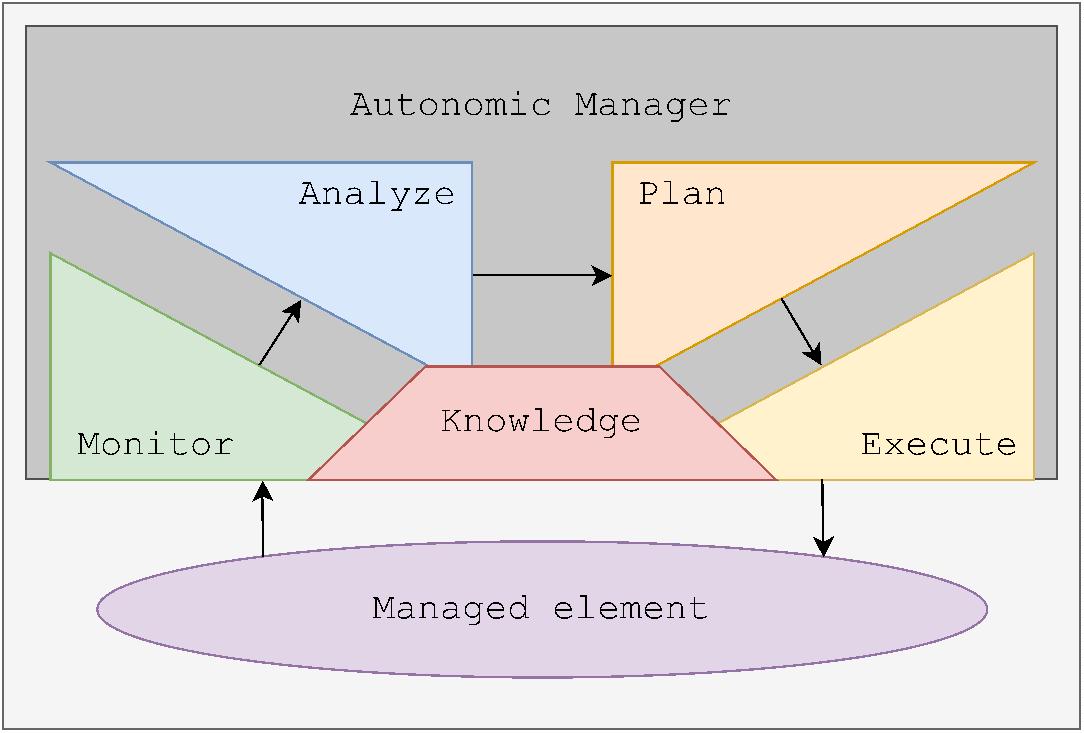
Planear (P): Durante la planiﬁcación se determina las acciones a tomar con el ﬁn

de llegar a un objetivo establecido a partir de una serie de reglas o estrategias.

Ejecutar (E): Finalmente, se ejecuta lo planeado usando los mecanismos

disponibles para el manejo del sistema.

5



4

4

4

.1.2 MECANÍSMOS DE DESCRIPCIÓN

.1.3 MECANÍSMOS DE ADAPTACIÓN

.2 SISTEMAS EMBEBIDOS

Los sistemas de cómputo embebidos, o simplemente sistemas embebidos, hacen

referencia a un sistema compuesto de microcontroladores los cuales están orientados a

llevar a cabo una función o un rango de funciones especíﬁcas (Heath, 2002). Este tipo

de sistemas, debido a la posibilidad de combinar hardware y software en una manera

compacta, se ha visto en multiples campos de la industria como lo son el sector

automotor, de maquinaria industrial o electrónica de consumo (Deichmann y cols.,

2

022).

4

4

4

.2.1 INTERNET OF THINGS

.2.2 SMART CAMPUS

.3 NOTACIÓN

El concepto de notación está deﬁnido como la representación gráﬁca del habla (Crystal,

011). En el contexto de las ciencias de la computación, esta idea se ha extrapolado

2

con el ﬁn de representar diferentes conceptos especíﬁcos del software y algoritmia de

manera visual (Rutanen, 2018). Esto puede verse con la existencia de lenguajes de

notación como lo es UML con el cual se realizan representaciones que van desde

arquitecturas de software, estructuras de base de datos, entre otros (Booch, Rumbaugh,

y Jacobson, 2005).

4

.3.1 GRAMÁTICA

La gramática, más especíﬁcamente gramáticas libres de contexto, son un conjunto de

reglas descriptivas. Este conjunto de reglas, en conjunto de una notación, cumplen la

función de dictar si una frase es válida para un lenguaje dado (Sipser, 2012, p. 101).

4

.3.2 SERIALIZACIÓN DE DATOS

La serialización de datos se reﬁere a la traducción de una estructura de datos hacia una

manera en la que pueda ser almacenada. En el contexto del proyecto, esta serialización

nos permitirá describir las arquitecturas objetivo a partir de la notación y gramáticas

6

establecidas. Así mismo, debería darnos la ﬂexibilidad de describir cualquier tipo de

meta para el sistema de software.

4

.4 ALGORITMIA DE COMPARACIÓN DE GRAFOS

7

5

METODOLOGÍA

Para el desarrollo del trabajo de grado, se propone un modelo de prototipado iterativo

compuesto de 5 fases (Ver ﬁg. 2). De esta manera, se avanzará a medida que se va

completando la fase anterior y permitirá a futuro el poder iterar sobre lo que se ha

desarrollado anteriormente.

Figura 2: Metodología del proyecto

5

.1 AMBIENTACIÓN CONCEPTUAL Y TECNOLÓGICA

La primera fase de la metodología se basa en la investigación de la literatura, al igual

que de la industria, necesaria para cubrir las bases tanto conceptuales como técnicas

necesarias para el desarrollo del proyecto.

ACTIVIDADES

5

5

5

.1.1. Identiﬁcación de las características principales de un sistema auto-adaptable.

.1.2. Análisis de los mecanismos de adaptación de la arquitectura.

.1.3. Análisis los algoritmos empleados para la comparación de la comparación de

las arquitecturas.

5

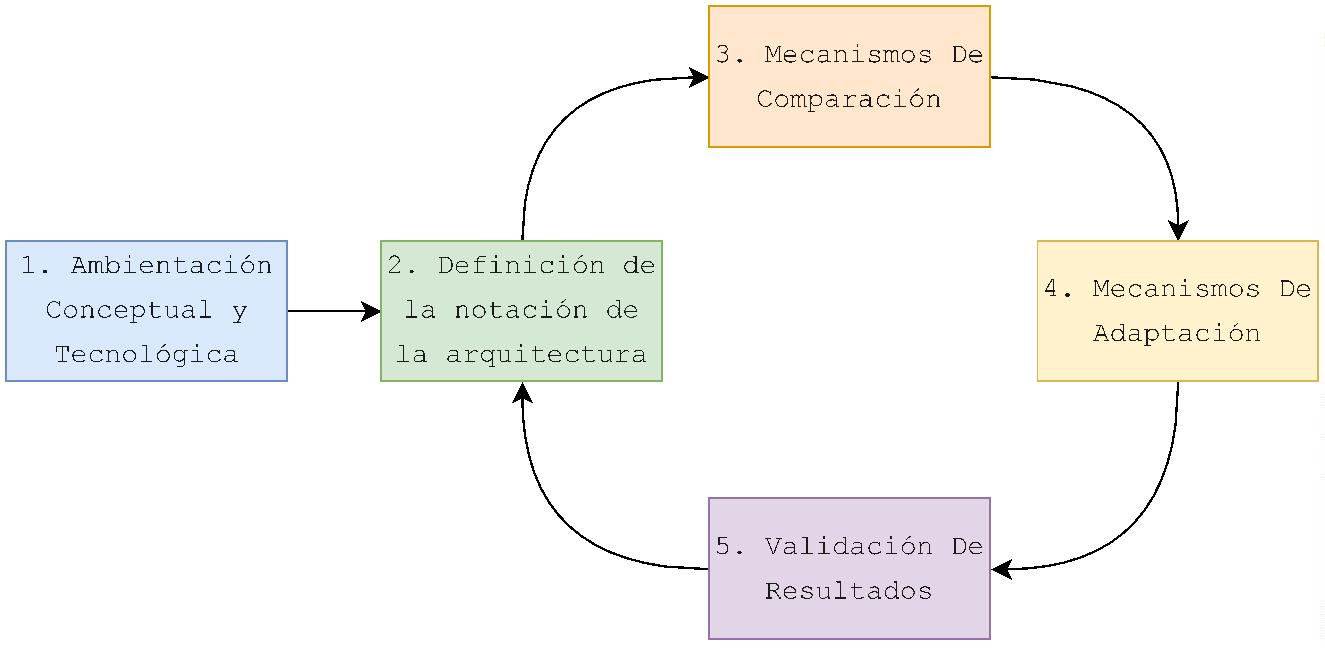
5

.1.4. Determinación de los criterios de selección para el lenguaje de notación.

.1.5. Evaluación de los posibles lenguajes de programación para la

implementación a realizar.

8



5

5

.1.6. Imprevistos.

.1.7. Análisis, retroalimentación y conclusiones del desarrollo de la fase.

5

.2 DEFINICIÓN DE LA NOTACIÓN DE LA ARQUITECTURA

La segunda fase está en la deﬁnición del cómo se realiza la declaración de la

arquitectura. Partiendo de los criterios de selección establecidos en la fase 1, se espera

determinar un lenguaje de notación el cual nos permita deﬁnir la arquitectura objetivo a

alcanzar, al igual que la gramática correspondiente para poder realizar dicha

declaración.

ACTIVIDADES

5

.2.1. Selección del lenguaje de marcado a usar a partir de los criterios

establecidos.

5

5

5

.2.2. Deﬁnición la gramática a usar para la deﬁnición de la arquitectura.

.2.3. Implementación la traducción de la notación al modelo de grafos.

.2.4. Determinación como se realizará la representación de los componentes y

partes de la arquitectura.

5

5

.2.5. Imprevistos.

.2.6. Análisis, retroalimentación y conclusiones del desarrollo de la fase.

5

.3 MECANISMOS DE COMPARACIÓN

Durante la tercera fase del proyecto, se buscará poder determinar e implementar cómo

se realizará la comparación entre el estado de la arquitectura obtenido durante la

auto-descripción de la misma y el objetivo establecido. Así mismo, y con el ﬁn de

reportar a los administradores de los sistemas, también será necesario deﬁnir niveles de

similitud entre las 2 arquitecturas.

ACTIVIDADES

5

5

.3.1. Selección del mecanismo de comparación a usar para evaluación de estado

de la arquitectura.

.3.2. Implementación del mecanismo de comparación seleccionado.

9

5

5

5

.3.3. Determinación de los diferentes niveles de similitud entre arquitecturas.

.3.4. Imprevistos.

.3.5. Análisis, retroalimentación y conclusiones del desarrollo de la fase.

5

.4 MECANISMOS DE ADAPTACIÓN

La cuarta fase del proyecto está orientada a la selección, al igual que la implementación

en Smart Campus UIS, del conjunto de mecanismos de adaptación de la arquitectura.

5

.4.1 ACTIVIDADES

5

5

5

5

.4.1. Deﬁnición el conjunto de mecanismos de adaptación.

.4.2. Implementación el conjunto de mecanismos de adaptación seleccionados.

.4.3. Imprevistos.

.4.4. Análisis, retroalimentación y conclusiones del desarrollo de la fase.

5

.5 VALIDACIÓN DE RESULTADOS

La fase ﬁnal del proyecto se encargará principalmente de la realización de pruebas de

los mecanismos implementados, los resultados obtenidos al igual que la documentación

de todo lo que se desarrolló durante el proyecto.

ACTIVIDADES

5

5

.5.1. Realización de las pruebas del funcionamiento de la implementación

realizada con diversas arquitecturas objetivo.

.5.2. Recopilación la documentación generada durante el desarrollo de cada una

de las fases del proyecto.

5

5

.5.3. Compilación de la documentación para generar el documento de ﬁnal.

.5.4. Correcciones y adiciones para la presentación ﬁnal del proyecto de grado.

1

0

6

CRONOGRAMA

Se debe realizar un cronograma que relacione las actividades prioritarias del proyecto y

el tiempo que destinará a cada una de ellas. Tenga en cuenta que el semestre tiene 16

semanas y debe desarrollar todo el trabajo de grado en este tiempo.

7

PRESUPUESTO

Descripción

DIRECTOR DE

PROYECTO

Responsable

UIS

Valor

COP

305.000/Hora

Cantidad

4 horas

mensuales por 4

meses

Precio

COP 4’880.000

PhD.

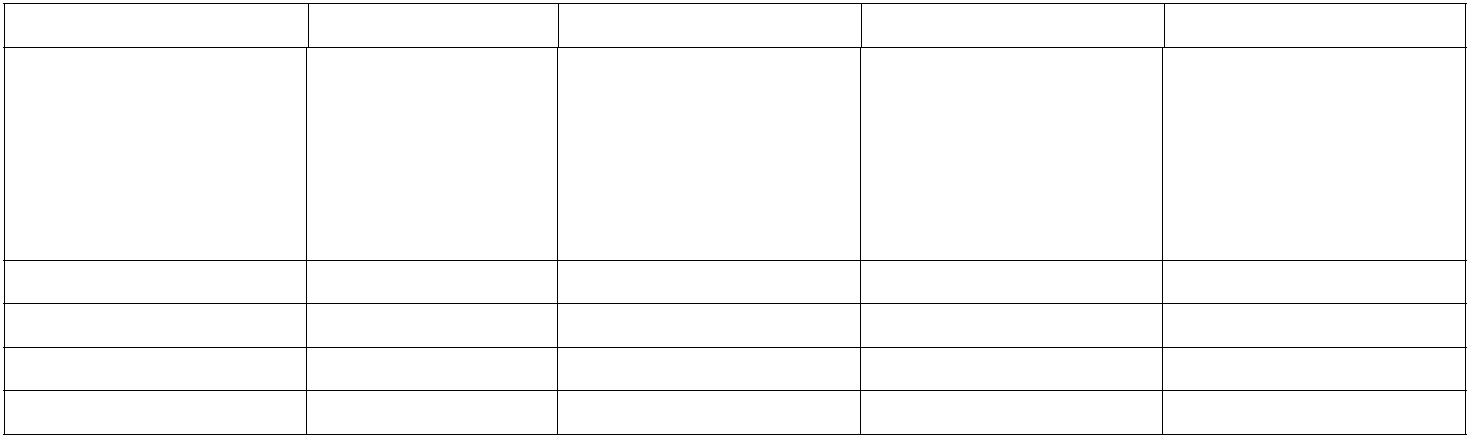
Gabriel

Rodrigo Pedraza

Ferreira

1

1



8

BIBLIOGRAFÍA

Booch, G., Rumbaugh, J., y Jacobson, I. (2005). The uniﬁed modeling language user

guide (2.a ed.). Boston, MA: Addison-Wesley Educational.

Crystal, D. (2011). A dictionary of linguistics and phonetics. John Wiley & Sons.

Deichmann, J., Doll, G., Klein, B., Mühlreiter, B., y Stein, J. P. (2022, Mar). Cracking the

complexity code in embedded systems development. McKinsey’s Advanced

Electronics Practice.

Forrester Research. (2019, Jun). Mainframe in the age of cloud, ai, and blockchain.

Forrester Consulting. Descargado de

https://www.ensono.com/resources/white-papers/old-workhorse-new-tech

-

mainframe-age-cloud-ai-and-blockchain-commissioned-study-conducted/

Gorla, A., Pezzè, M., Wuttke, J., Mariani, L., y Pastore, F. (2010, 01). Achieving

cost-effective software reliability through self-healing. Computing and Informatics,

2

9, 93-115.

Heath, S. (2002). Embedded systems design. Elsevier.

Horn, P. (2001, Oct). autonomic computing: Ibm’s perspective on the state of information

technology. IBM. Descargado de https://homeostasis.scs.carleton.ca/~soma/

biosec/readings/autonomic\_computing.pdf

Jiménez, H., Cárcamo, E., y Pedraza, G. (2020). Extensible software platform for smart

campus based on microservices. RISTI - Revista Iberica de Sistemas e

Tecnologias de Informacao, 2020(E38), 270-282. Descargado de www.scopus.com

Kephart, J. (2011, 01). Autonomic computing: the ﬁrst decade. En (p. 1-2). doi:

1

0.1145/1998582.1998584

Krikava, F. (2013, 11). Domain-speciﬁc modeling language for self-adaptive software

system architectures.

Lalanda, P., Diaconescu, A., y McCann, J. A. (2014). Autonomic computing: Principles,

design and implementation. Springer.

Loukides, M. (2021). The cloud in 2021: Adoption continues. Descargado de

https://get.oreilly.com/rs/107-FMS-070/images/

The-Cloud-in-2021-Adoption-Continues.pdf?mkt\_tok=

MTA3LUZNUy0wNzAAAAGISYNxeMWRA\_a\_GPKBEqQliGws2SImdqefJ4Ch11jEKmmSuN

\_

ccGOOgoUv9enxj\_0pbnchAdPjkL3QgDEdY4Xf5j\_teuCKfiXTQIdg2jy7ETKmudbu

McCann, J. A., y Huebscher, M. C. (2004). Evaluation issues in autonomic computing.

En H. Jin, Y. Pan, N. Xiao, y J. Sun (Eds.), Grid and cooperative computing - gcc

2

004 workshops (pp. 597–608). Berlin, Heidelberg: Springer Berlin Heidelberg.

Rutanen, K. (2018). Minimal characterization of o-notation in algorithm analysis.

Theoretical computer science, 713, 31–41.

Sipser, M. (2012). Introduction to the theory of computation (3.a ed.). Belmont, CA:

Wadsworth Publishing.

1

2