ALGORITMO NOVEDOSO PARA LA DETECCION DE TAREAS REPETITIVAS EN EL TECLADO

NOVEL ALGORITHM FOR DETECTION OF REPETITIVE TASKS IN THE KEYBOARD

Recibido: 3 de marzo 2015 - aceptado: 25 de mayo 2015

Bairon Londoño González[1]
Universidad Simón Bolívar

Paola Andrea Sánchez[2]
Universidad Simón Bolívar

Keywords:

Automation of repetitive tasks, algorithms for detecting patterns, keyboard commands, automata.

Abstract

In this paper a tool for the detection of repetitive tasks with logical sequences realized across command of the keyboard is proposed, by means of the design and implementation of an algorithm based on the use of finite automata deterministic and agents of bosses' search. The innovation of the developed algorithm takes root in that it is orientated to the detection of repetitive tasks which activities have a logical sequence and that nowadays are not automated by the complex thing that is this labor.

Palabras clave:

Automatización de tareas repetitivas, algoritmos para la detección de patrones, comandos de teclado, autómatas.

Resumen

En este artículo se propone una herramienta para la detección de tareas repetitivas con secuencias lógicas realizadas a través de comandos del teclado, mediante el diseño e implementación de un algoritmo basado en el uso de autómatas finitos determinísticos y agentes de búsqueda de patrones. La novedad del algoritmo desarrollado radica en que está orientado a la detección de tareas repetitivas cuyas actividades tienen una secuencia lógica y que actualmente no se encuentran automatizadas por lo complejo que es esta labor.

1. Maestría en Ingeniería de Sistemas y Computación. Docente del Departamento de Ciencias básicas de la Universidad Simón Bolívar. e-mail: blondono@unisimonbolivar.edu.co.

2. Doctora en Ingeniería- Área Ingeniería de Sistemas. Miembro del grupo de investigación Ingebiocaribe, Universidad Simón Bolívar. E-mail: pasanchez9@unisimonbolivar.edu.co

*Este artículo es asociado al a la tesis de maestría llamada algoritmo novedoso para la detección de tareas repetitivas en el teclado

1 INTRODUCCIÓN

Desde los inicios de los sistemas operativos de computadoras se han realizado modificaciones que giran en torno a hacer más fácil el trabajo a los usuarios; cambios como la creación de interfaces amigables, programas y funciones que sistematicen procesos, nos ayudan a realizar con facilidad las tareas más comunes; sin embargo, existe infinidades de tareas que son tediosas para llevar a cabo por los usuarios debido a que son repetitivas y su ejecución suele tomar mucho tiempo.[1]

Las tareas repetitivas en el computador se pueden clasificar en tres tipos: Tareas con operaciones repetitivas fijas, tareas repetitivas cuyas operaciones tienen una sucesión lógica y tareas cuyas operaciones no siguen una sucesión lógica.[2] Para la primera clasificación existe una amplia variedad de herramientas que apoyan la labor del usuario, realizando dichas tareas de forma automática, y con el fin de facilitar su ejecución y eficiencia; sin embargo, para los dos conjuntos de tareas restantes no existen herramientas que asistan directamente al usuario, esto quizá por la dificultad que tiene su automatización.

A nivel comercial, es posible encontrar numerosas soluciones de software que prometen al usuario una automatización de sus tareas, incluso incluyendo, al menos en definición el segundo tipo [3]; no obstante, su ejecución dista mucho de lo esperado. La concentración de desarrollos en el campo comercial, conduce a que sean escasos los trabajos que documenten este tipo de innovaciones, dificultando la tarea de los investigadores académicos en el campo.

Ahora bien, el proceso de automatización de tareas en el teclado por parte del sistema implica la ejecución de varias etapas independientes: detección de la tarea, aprendizaje de los movimientos realizados por el usuario, predicción de las acciones que realizará y sugerencia de acciones. Diversos algoritmos han sido propuestos en la literatura para la ejecución de cada etapa, donde los basados en el uso de autómatas y agentes de decisión han demostrado ser eficientes en la detección y aprendizaje, aunque presentan fallas en la predicción y sólo se enfocan en tareas repetitivas fijas [4][5][6].

Además, otro aspecto que dificulta la detección de tareas repetitivas, es la carencia de una clasificación profunda de los comandos de teclado de Windows, identificando que comandos pertenecen a tareas repetitivas o incluso que comandos permiten el inicio, final o hacen parte del cuerpo de actividades de tareas repetitivas.[7]

Los problemas aquí planteados acerca de la detección de tareas repetitivas evidencian la necesidad de contar con herramientas adecuadas que permitan una identificación acertada y eficaz de dichas labores. En consecuencia, el tópico principal de este artículo es el desarrollo de un algoritmo novedoso para la detección de tareas repetitivas con secuencia lógica, y su validación a través de experimentos artificiales y reales. La representación del problema se basa en el uso de autómatas finitos deterministas, los cuales permiten una acertada comprensión de las características del problema.

2 AUTOMATIZACIÓN DE TAREAS

Tal como lo definen Kaber & Prinzel [8], automatización se refiere a "sistemas o métodos en los cuales muchos de los procesos son realizados automáticamente o son controlados por máquinas autónomas o dispositivos electrónicos". La automatización es una herramienta, o un recurso, que el usuario humano puede usar para ayudar a realizar alguna tarea tediosa o imposible sin la ayuda de la máquina [9]. Por lo tanto, se puede considerar la automatización como un proceso de sustitución parcial o total de la actividad del usuario humano por algún dispositivo o máquina [10]. Otros autores definen la automatización de tareas, en informática, como el conjunto de métodos que sirven para realizar tareas repetitivas en un computador.

Automatizar implica, básicamente, que el hombre no intervenga, o lo haga muy limitadamente, en la ejecución de tareas; en otras palabras, que el computador tenga autonomía en la sugerencia, decisión y ejecución de tales acciones. [11]

En la búsqueda de facilitar lo complejo, la comunidad académica se ha volcado en el estudio del tipo de tareas que podrían realizarse “de mejor forma” por un sistema automatizado; esto es, tareas que pudiéndola realizar un humano, la eficacia y precisión puede ser mayor cuando la realiza un computador.

En la ejecución de tareas automáticas interviene el usuario humano y el computador, y su rol, tal como lo afirman [12], puede ser clasificado de acuerdo a la cantidad de carga de trabajo que realiza el computador en comparación con lo que el humano hace por sí solo (Ver Figura 1). El computador puede extender las capacidades del humano, puede reducir parcialmente su trabajo haciéndolo más fácil, realizar copias de

seguridad o respaldo de su trabajo en caso que falte, o sustituir al humano completamente.[13] Cuando el computador y el humano están trabajando en la misma tarea al mismo tiempo, se conoce como control compartido. Cuando trabajan en la misma tarea en diferentes momentos se trata de control de intercambio.

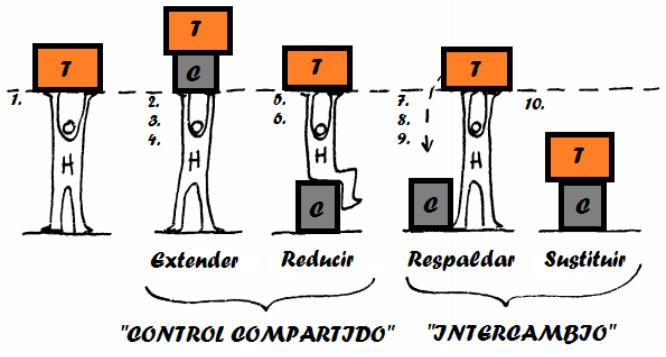


Figura 1. Roles Humano vs Computador en la ejecución de tareas. [12]

2.1 Clasificación de tareas repetitivas en el teclado

En el contexto de este artículo, una actividad es un conjunto de comandos cuya finalidad es realizar una transacción u operación; por ejemplo, cortar y pegar un elemento, actualizar un elemento, eliminar, etc. y una tarea es un conjunto de actividades que realiza el usuario con la interacción del teclado. [14]

Ahora bien, las tareas que se realizan con frecuencia en el teclado, pueden ser clasificadas según la complejidad que se requiera para su ejecución, de la siguiente manera:

2.1.1 Tareas con actividades repetitivas fijas

Representan aquellas tareas cuyas actividades se repiten siempre de forma exacta, es decir, utilizando los mismos comandos en su ejecución. Este tipo de tareas han sido ampliamente estudiadas en la literatura, y existen múltiples herramientas que permiten su identificación y la realización automática, tales como Batch®, Autohotkey®[1], Macro Recorder®, Automator®, entre otros.[15]

Un ejemplo de este tipo de tarea se presenta en la **¡Error! No se encuentra el origen de la referencia.**, la cual representa la tarea de copiar elementos de un software a otro, en el orden exacto. En el ejemplo los elementos de la primera fila del Software 1 deben ser copiados a la primera columna del Software 2, siguiendo exactamente el orden de los elementos.

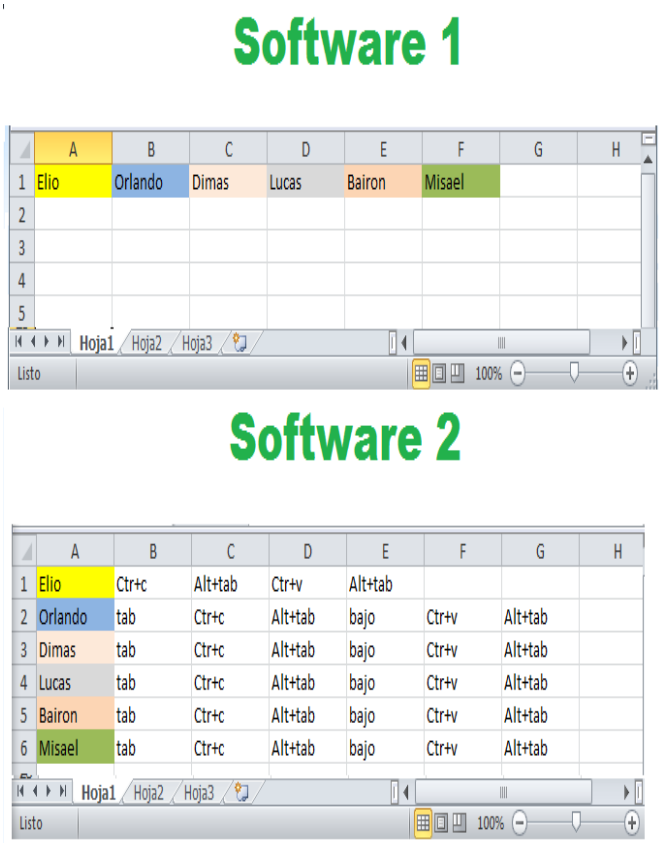


Figura 2. Tareas con actividades repetitivas fijas

En el lado derecho de cada uno de los elementos copiados en el software 2 de la **¡Error! No se encuentra el origen de la referencia.**, se muestran los comandos del teclado utilizados para realizar cada operación. Como se puede observar cada renglón equivale a una actividad, y los comandos usados en cada una de ellas son iguales, exceptuando la actividad inicial.

2.1.2 Tareas repetitivas cuyas actividades tienen una sucesión lógica:

Representan aquellas tareas cuyas actividades se repiten siguiendo una secuencia lógica, con los mismos comandos de inicio y fin, pero que varían en los comandos que componen el cuerpo de la actividad. Este tipo de tareas representan una dificultad mayor que el caso simple, y las herramientas desarrolladas para el primer caso no permiten su identificación, y por lo tanto, no son adecuadas para su automatización.[16]

Como ejemplo de estas tareas en la **¡Error! No se encuentra el origen de la referencia.** se presenta el proceso de copiar elementos de un software a otro, en

[1] Disponible para descarga en http://www.autohotkey.com/

un orden predeterminado; los elementos de la primera fila del Software 1 deben copiarse en la primera columna del Software 2, siguiendo un orden secuencial, pero incrementando de forma lógica los espacios entre las celdas.

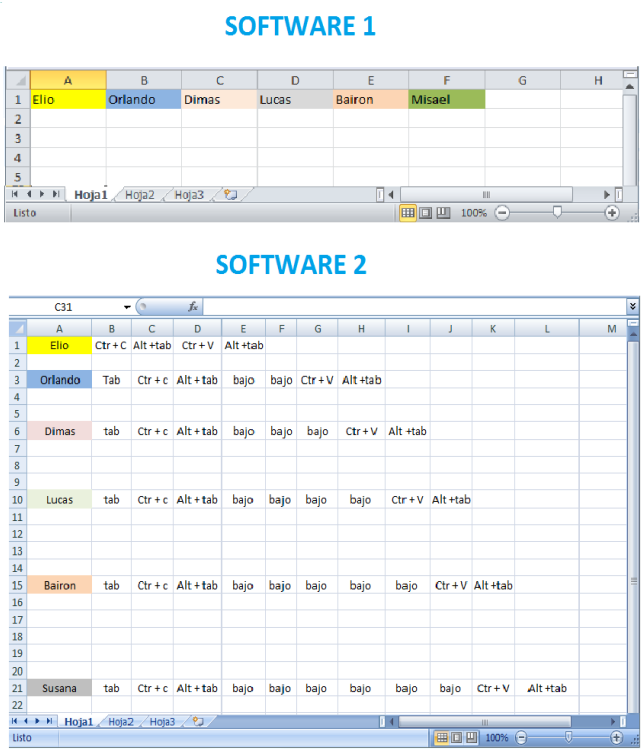


Figura 3. Tareas repetitivas cuyas actividades tienen una sucesión lógica

Como puede observarse en el lado derecho del software 2 los comandos usados para realizar cada actividad, exceptuando la actividad inicial, coinciden en el inicio y final, y los comandos del cuerpo de la actividad siguen una estructura lógica de comandos que se repiten.[17][18]

2.1.3 Tareas cuyas actividades no tienen una sucesión lógica:

Representan aquellas tareas cuyas actividades se repiten siguiendo una secuencia no lógica, donde, aunque no coinciden en su orden los comandos de inicio, fin, y el cuerpo de la actividad, se realiza una actividad repetitiva en la medida que hay presencia de los comandos principales de ejecución de la actividad. Igual que en el caso de tareas repetitivas con secuencia lógica, para tareas con secuencias no lógicas las herramientas disponibles no permiten su identificación, y no son adecuadas para su automatización.

Un ejemplo de tareas repetitivas sin secuencia lógica es mostrado en la **¡Error! No se encuentra el origen de la referencia.**; en esta se presenta la tarea de copiar elementos de un software a otro, en un orden no lógico; los elementos de la primera fila del Software 1 deben copiarse en la primera columna del Software 2, en un orden que no sigue ninguna secuencia lógica de ejecución.[19]

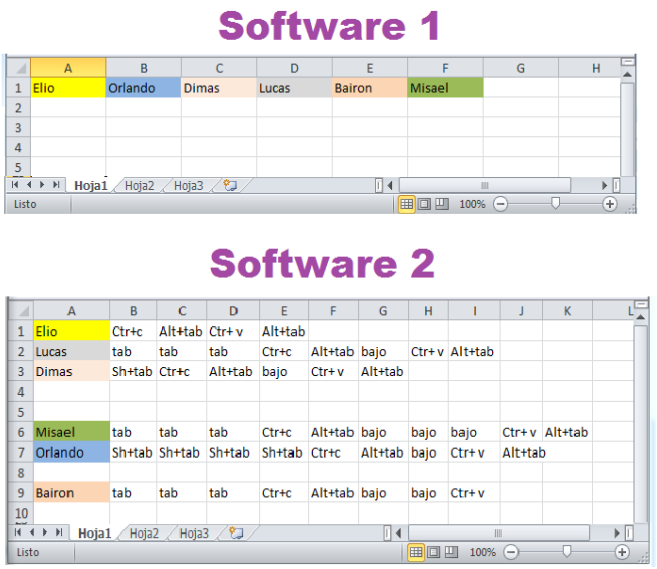


Figura 4. Tareas cuyas actividades no tiene una sucesión lógica

En el lado derecho del software 2, se observa que los comandos usados para realizar cada actividad no coinciden en su estructura, sin embargo, un análisis mayor de las actividades permite observar que comandos como Ctrl+c, Alt+Tab y Ctrl+v, se repiten en todas las actividades, aunque no en un orden lógico.[20]

2.2 Evolución de la automatización de tareas repetitivas en el computador

El ingreso de secuencias repetitivas de comandos (o tareas repetitivas) es una característica bien conocida de la interacción humano-computador. Los primeros trabajos desarrollados entorno al problema de automatizar dichas tareas se asocian con las conocidas macros o comandos programados para la realización de ciertas tareas. Dichos elementos le permiten al usuario escribir un programa que se puede invocar y que realiza una secuencia de comandos automáticamente. El limitante que presentan estos acercamientos es que, en general, los usuarios no quieren o no pueden dedicar demasiado esfuerzo a la programación; para usuarios inexpertos y con escasos conocimientos de programación, escribir un programa, a menudo, toma más tiempo que la realización de una secuencia manual de comandos. [21]

La investigación acerca de la automatización de tareas repetitivas en el teclado se ha volcado a superar las limitaciones enunciadas desde diferentes campos de

investigación: programación por demostración (*programming by demonstration* - PBD), interfaces predictivas (*predictive interfaces*) y agentes de aprendizaje de interfaz (*learning interface agents*).

Los sistemas PBD [22] le permiten al usuario demostrar las tareas que quiere que sean automatizadas y, a partir de esto, crear un programa. La grabación de macros son los primeros ejemplos de sistemas PBD, pero se limitaban a que los comandos grabados debían ser muy específicos para ser reutilizado (aprendizaje de memoria, sin parametrización). Sistemas PBD más sofisticados como Mondrian [23], permiten crear programas a partir de las acciones del usuario que contienen variables, bucles iterativos o condicionales. A pesar que el uso de sistemas PBD no requiere conocimientos de programación, toda vez que el usuario no requiere escribir el código, el hecho de realizar la "demostración" lleva tiempo, e interrumpe el flujo de trabajo del usuario.

Las interfaces predictivas [18] y los agentes de aprendizaje de interfaz [24] [25] se basan en observar las acciones que el usuario realiza mientras manipula el computador. Dichas técnicas procuran aprender de las correlaciones entre las situaciones que el usuario emprende y los correspondientes comandos que ejecuta, para así predecir el próximo comando que ejecutará. Posteriormente, estos sistemas asisten al usuario en la ejecución de los comandos predichos, sugiriendo algunos comandos que se llevaran a cabo automáticamente, y a su vez pueden recibir retroalimentación del usuario o ser entrenados con base en ejemplos hipotéticos. Las interfaces predictivas se enfocan en identificar las acciones repetitivas del usuario y predecir qué hará, mientras que los agentes de interfaz se focalizan en la asistencia al usuario. La Tabla 1, presenta algunos ejemplos de los sistemas PBD, de interfaces predictivas y de agentes de aprendizaje de interfaz más usados en la literatura.[26]

La principal desventaja de los sistemas presentados en la Tabla 1, es que se enfocan en tareas repetitivas fijas, haciendo a un lado tareas más complejas como son las repetitivas con secuencias lógicas, siendo pequeño y pre-establecido el conjunto de acciones que la mayoría de estos sistemas sugiere.[32]

2.3 Proceso de automatización de tareas repetitivas en el computador

El proceso automatización de tareas repetitivas en el teclado implica la realización de una serie de etapas:

i. Realización de unas tareas repetitivas iniciales (Usuario)
ii. Identificación, en tiempo real, que el Usuario está realizando una tarea repetitiva (Computador)

Tabla 1. Resumen de aplicaciones orientadas a la automatización de tareas repetitivas

Sistema	Tipo	Descripción
AutoHotkey	PBD	Asistente para automatizar órdenes ingresadas por teclado y ratón a través de macros.
Automator[27]	PBD	Asistente de procesos de automatización para Mac OSX que facilita la ejecución de tareas repetitivas fijas en diferentes aplicaciones y partes del sistema operativo OS X.
Cron, Crontab, Grontab, Kcron, Gnome-Schedule	PBD	Asistentes que permiten programar tareas repetitivas fijas para que se ejecuten bajo Linux
AnyTask	PBD	Permite la automatización de algunas tareas repetitivas fijas
AutomationBatch Tools	PBD	Permite la automatización y el procesamiento por lotes de tareas repetitivas fijas
JitBit Macro Recorder LITE	PBD	Asistente para la grabación de acciones simples de teclado y ratón.
Automatización	PBD	Herramienta para la creación de reglas de tareas repetitivas fijas
RoboTask[2]	PBD	Permite automatizar tareas comunes, tales como revisar correo, mover, almacenar, cargar o descargar archivos y enviarlos por correo.
WinAutomation[3]	PBD	App de Windows que permite la grabación de macros y de web
CAP [28]	predictivo	Asistente para la gestión de calendarios de reuniones.
ClipBoard[29]	predictivo	Interfaz para Unix que intenta predecir los comandos que el usuario va emitir.
OpenSesame! [30]	Agentes y predictivo	Asistente para Macintosh 7 que ofrece la realización de tareas comunes como abrir o cerrar archivos o aplicaciones, vaciar la papelera de reciclaje, etc.
Reactive Keyboard [18]	predictivo	Asistente que predice el texto que el usuario escribirá tras comenzar a escribir las palabras.
WebWatcher [31]	predictivo	Asistente para de navegación web que sugiere enlaces de interés para el usuario.
Eager [22]	PBD, agentes y predictivo	Asistente para Macintosh que cuando detecta dos ocurrencias consecutivas de una tarea repetitiva fija, supone que son las dos primeras iteraciones de un bucle y se propone completar el ciclo.
[24]	agentes	Asistente para el manejo de correo electrónico, programación de agenda, y el filtrado de noticias de prensa.
APE [7]	agentes	Asistente que observa las acciones de los usuarios, aprende sus hábitos, y realiza sugerencias de

[2] http://www.robotask.com/

[3] http://www.winautomation.com

		acciones para completar tareas repetitivas.

Fuente: Recopilación propia

iii. Aprendizaje de las tareas repetitivas que el usuario realiza (Computador)
iv. Predicción de las labores que el usuario hará (Computador)
v. Sugerencia de las labores que debería realizar posteriormente el usuario (Computador)
vi. Aceptación o rechazo de sugerencia de acciones (Usuario)

Cada una de las etapas mencionadas implica, a su vez, una serie de procesos para llevarla a cabo, por lo tanto, se pueden identificar problemas no resueltos en cada una de ellas. [33]

Ahora bien, la automatización de tareas es una rama proveniente del amplio campo del aprendizaje de máquinas. En la automatización es común el uso de algoritmos supervisados o predictivos que permiten aprender y extraer conocimiento de dichas tareas. Estos algoritmos predicen el valor de un atributo, a partir de otros atributos conocidos. Se dividen en: Algoritmos que resuelven problemas de clasificación, tales como árboles y tablas de decisión, algoritmos evolutivos, entre otros; y algoritmos que se utilizan en la predicción, como las redes neuronales.

Como se vio anteriormente, por parte del computador, la automatización de tareas repetitivas implica la ejecución de cuatro etapas independientes, donde cada proceso requiere un tratamiento individual y con orientaciones diferentes.

i. La identificación de tareas repetitivas: es equivalente a las etapas adquisición de datos y extracción de características de un proceso de reconocimiento de patrones, y deja para un proceso posterior la clasificación de las tareas; para detectar la tarea repetitiva, desde la literatura, se hace uso de técnicas como: Árboles y Tablas de decisión, Algoritmos genéticos, Heurísticos y Redes neuronales [34]
ii. El aprendizaje de las acciones del usuario: abarca el proceso implícito en la clasificación o caracterización de las tareas previamente detectadas y la actualización de la base de conocimiento. Las técnicas usadas para el aprendizaje de las acciones son: Aprendizaje de conceptos, Aprendizaje por refuerzo, Aprendizaje basado en casos, Programación por demostración, Programación por ejemplos, Redes neuronales, clasificación bayesiana, *clustering*, análisis de componentes principales, entre otros.[35]
iii. En la predicción de las acciones que el usuario realizará se utilizan algoritmos de predicción. Bajo esta clasificación las técnicas más representativas son: Redes neuronales, Programación no lineal, Agentes predictivos, Sistemas dinámicos, Algoritmos difusos, entre otros.
iv. La asistencia o sugerencia de las acciones a realizar implica la interacción del sistema con el usuario, bien sea bajo asistencia dirigida (cuando el usuario dice que acciones realizar), asistencia controlada (cuando el sistema sugiere las acciones y el usuario decide si realizarlas o no), y asistencia automática (cuando el sistema decide y ejecuta las acciones sin requerir aprobación).Algunas técnicas de asistencia en automatización de tareas en el computador incluyen los llamados Agentes de interfaz y Agentes de cooperación.

Este trabajo se orienta al desarrollo de las etapas i y ii, cuando las tareas desarrolladas siguen una secuencia repetitiva lógica no trivial, y se limita a la identificación de acciones realizadas a través de comandos de teclado.[36]

3 PROTOCOLO PARA LA ESPECIFICACIÓN Y CONSTRUCCIÓN DE UN ALGORITMO PARA LA DETECCIÓN DE TAREAS REPETITIVAS

A continuación se presenta el protocolo seguido para el diseño y construcción de un algoritmo para la detección de tareas repetitivas realizadas a través comandos de teclado:

3.1 Clasificación de comandos

En esta etapa se seleccionan los comandos de teclado susceptibles de ser usados en la realización de una tarea repetitiva, y se clasifican estos según su nivel, tipo y uso:

i. Se toma la clasificación general de comandos de Windows separada por entornos y se realiza un análisis de los comandos que aplican en tareas repetitivas. [37]
ii. basados en el proceso previo de clasificación, se seleccionan los comandos finales que hacen parte del proceso de una tarea repetitiva simple o con secuencia lógica. La **Tabla 2**, presenta el resumen de la clasificación realizada de los comandos de Windows que si aplican en tareas repetitivas con un nivel de importancia Medio y Alto, en la cual, para cada comando, se muestra su Nivel, Tipo y Uso. En esta tabla se tiene un total de once comandos de Windows con un nivel Alto y once con un nivel Medio. Además, se tienen dos comandos de tipo

Inicial, uno de tipo Final, tres de tipo Inicial-Final y dieciséis de tipo Normal. Finalmente, se encuentran doce de uso Interno, seis de uso Externo y cuatro de uso Interno-Externo.

3.2 Extracción de reglas

En este proceso se establece el orden en el cual tienen que aparecer los comandos seleccionados para formar una actividad de una tarea repetitiva. Ejemplo si un tarea repetitiva inicia con un copiado (CTR+C) el siguiente paso será realizar un movimiento hacia el destino del pegado (TAB, ALT+TAB, etc.) ya ubicado el destino se termina la actividad con un pegado (CTR+V). El proceso finalizó con una identificación total de 81 reglas de pares de comandos.[38]

Tabla 2. Clasificación final de comandos de teclado en Windows

Entorno	Comando	Nivel	Tipo	Uso
Generales	Ctrl+C (o Ctrl+Insert)	Alto	Inicial	Interno
Generales	Ctrl+X	Alto	Inicial	Interno
Generales	Ctrl+V (o Mayus+Insert)	Alto	Final	Interno
Generales	Suprimir (o Ctrl+D)	Medio	Inicial-Final	Interno
Generales	Mayus+Suprimir	Medio	Inicial-Final	Interno
Generales	F2	Medio	Inicial-Final	Interno
Generales	Mayus + FD	Alto	Normal	Interno
Generales	Ctrl+A	Medio	Normal	Interno
Generales	Alt+Tab	Alto	Normal	Externo
Generales	Ctrl+Alt+Tab	Medio	Normal	Externo
Generales	Windows + Tab	Alto	Normal	Externo
Generales	Ctrl+Windows +Tab	Medio	Normal	Externo
Generales	F6	Alto	Normal	Interno
Generales	Alt+Letra	Medio	Normal	Interno
Cuadros de diálogo	Tab	Alto	Normal	Interno-Externo
Cuadros de diálogo	Mayus+Tab	Alto	Normal	Interno-Externo
Cuadros de diálogo	Enter	Alto	Normal	Interno-Externo
Cuadros de diálogo	FD	Alto	Normal	Interno-Externo
logotipo de Windows	Windows +Numero	Medio	Normal	Externo
logotipo de Windows	Ctrl+Windows +Numero	Medio	Normal	Externo
Explorador de Windows	Fin	Medio	Normal	Interno
Explorador de Windows	Inicial	Medio	Normal	Interno

Fuente: Construcción propia.

3.3 Diseño del Autómata

A partir de los comandos seleccionados teniendo en cuenta su Tipoy las relaciones expresadas en las 81 reglas, se construyó el autómata finito determinista (AFD) que representa nuestra base de conocimiento. El AFD cuenta con 19 Nodos (Estado o posición en un Autómata) representados con la letra "q" y con 81 relaciones representadas con flechas de dirección que pueden ser dirigidas hacia otro Nodo o al mismo Nodo; el AFD comienza con un Nodo de inicio (q0) el cual está relacionado con los comandos que se pueden utilizar para iniciar una tarea repetitiva (Comandos tipo Inicial, Inicial-Final), posteriormente, q0 nos lleva a Nodos (q1,..., q6, q8,..., q12, q14,...., q17) que pertenecen al cuerpo de la tarea repetitiva (Comandos de tipo Normal); finalmente, una actividad culmina en un Nodo final (q7,q13,q18) mediante un comando tipo Final o Inicial-Final. En la **¡Error! No se encuentra el origen de la referencia.** se presenta el esquema del AFD diseñado.

3.4 Diccionario de comandos

A partir de los datos condensados en la **Tabla 2**, se construyó el diccionario de comandos a utilizar en el autómata, el cual es presentado en la **Tabla 3**. Este diccionario es construido para la representación del AFD en una tabla de transición que facilitara posteriormente la codificación del autómata en un lenguaje de programación. El diccionario consiste en relacionar cada comando de Windows con un número identificador.

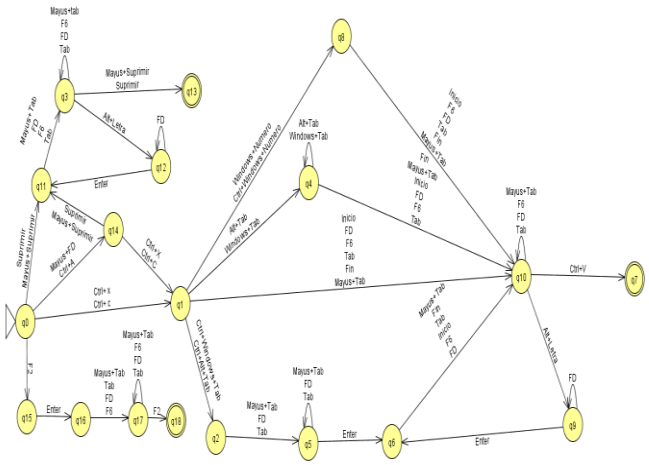


Figura 5. Autómata Finito Determinista para la detección de tareas repetitivas con secuencia lógica en sus actividades

3.5 Métodos ejecutados por el Agente para la detección de tareas repetitivas

Como se enunció anteriormente, el Agente juega el rol principal en el sistema desarrollado para la detección de tareas repetitivas. A continuación se presenta en detalle los procesos realizados por el agente. [39] [40

Tabla 3. Diccionario de comandos seleccionados

Numero	Comando	Numero	Comando
0	Ctrl+ C	11	Enter
1	Ctrl+ X	12	Alt+Letra
2	Mayus+FD	13	Windows+Numero
3	Ctrl+A	14	Ctrl+Windows+Numero
4	Mayus+Suprimir	15	Alt+Tab
5	Suprimir	16	Windows+Tab
6	F2	17	Ctrl+Windows+Tab
7	Tab	18	Ctrl+Alt+Tab
8	F6	19	Fin
9	FD	20	Inicial
10	Mayus+Tab	21	Ctrl+v

Fuente: Construcción propia

3.5.1 Evaluar comandos

Los comandos que son recibidos del KeyLogger y son catalogados como posibles entradas, por hacer parte del diccionario de comandos, son remitidos al AFD, para que éste último lo evalúe. Cada comando es categorizado por el AFD, y la evaluación es entregada al Agente. La evaluación de los comandos realizada por el AFD puede ser de las siguientes tres formas:

i. Aceptado: quiere decir que el comando evaluado es el inicio de una actividad o pertenece al cuerpo de la actividad en ejecución; cuando el Agente recibe esta respuesta agrega el comando a la actividad actual o la reinicia en caso de que el comando evaluado sea el inicio de otra actividad excluyente a esta.

ii. Final: quiere decir que el comando evaluado permitió la finalización de una actividad; cuando el Agente recibe esta respuesta agrega la actividad a la cola de la tarea.

iii. Negado: quiere decir que el comando evaluado es característico de una tarea repetitiva pero no es admitido en el estado actual del AFD; cuando el Agente recibe esta respuesta reinicia la actividad en ejecución.[41]

Algorítmicamente el proceso de evaluación de comandos mencionado puede representarse como:

Algoritmo 1. Algoritmo evaluador de comandos en el AFD

```
Método evaluarAFD(comando)
1:  if (comando_tipo = inicial) and (actividad = vacio)then
2:      cambiarPosicionAFD(comando)
3:      actividad.crear()
4:      actividad.agregar(comando)
5:  elseif (comando_tipo=inicial) and (actividad=!vacio) then
6:      actividad.reiniciar()
7:      AFD.reiniciar()
8:      evaluarAFD(comando)
9:
    elseif(comando_tipo=normal)andevaluarComando(comando)=Aceptado then
10:     cambiarPosicionAFD(comando)
11:     actividad.agregar(comando)
12:
    elseif(comando_tipo=final)andevaluarComando(comando)=Aceptado then
13:     cambiarPosicionAFD(comando)
14:     actividad.agregar(comando)
15:     tarea.agregar(actividad)
16:     actividad.reiniciar()
17:     detectarTarea()
18: else
19:     reciclar(comando,tipo_negado)
20: endif
```

El algoritmo comienza probando las diferentes instancias del comando: Aceptado (líneas del 1 al 11 del

Algoritmo 1), Final (líneas del 12 al 17 del

Algoritmo 1) Negado (líneas del 18 al 20 del

Algoritmo 1). En el estado Aceptado, como se mencionó anteriormente, puede ocurrir:

- que el comando sea el inicio de una actividad (AFD clasifica el comando como tipo inicial): caso en el cual, se crea una actividad, si esta es el comienzo de la tarea [líneas 1-4]; o se reinicia la actividad, si ya hay una actividad actual [líneas 5-8].
- que el comando haga parte del cuerpo de una actividad en ejecución, en este caso la categoría arrojada por el AFD para el comando es tipo normal, por lo tanto, el agente agrega dicho elemento a la cola de la actividad actual [líneas 9-11].

Cuando el AFD clasifica el comando como tipo Final, el Agente agrega el comando a la cola de las actividades, a

su vez, agrega la actividad a la cola de tareas, reinicia la actividad y realiza la validación de si corresponde a una tarea repetitiva (Ver posteriormente en procedimiento correspondiente al Algoritmo 2).
Cuando el AFD clasifica el comando tipo Negado, el Agente ejecuta el procedimiento correspondiente al algoritmo reciclado de comandos de tipo Negado (ver posteriormente en procedimiento correspondiente al Algoritmo 4).

3.5.2 Detectar tarea repetitiva

Los comandos recibidos del Keylogger y aceptados por el AFD, son agrupados en actividades[4]. El agente utiliza un algoritmo basado en las últimas tres actividades agregadas en la cola de la tarea y evalúa las características de tales actividades. El agente puede detectar si se está ejecutando una tarea repetitiva o no, cuando cumple a su vez 4 condiciones: la cantidad de actividades acumuladas en cola es mayor o igual a tres, cada actividad tiene el mismo comando tipo Inicial [se evalúa en mediante el proceso analizarInicioActividades()], el mismo comando tipo Final [se evalúa en mediante el proceso analizarFinalActividades()] y una sucesión de comandos tipo Normal válidos para el AFD [se evalúa en mediante el proceso analizarCuerpoActividades()]. En caso de detectar una tarea repetitiva le informara de inmediato al usuario. A continuación se representa algorítmicamente el procedimiento mencionado.[42]

Algoritmo 2. Algoritmo detector de tareas repetitivas
Método**detectarTarea**()

```
1:      if(tarea.cantidad() >=3)
2:          if(analizarInicioActividades()=true)and
(analizarFinalActividades()=true) and
(analizarCuerpoActividades()=true)then
3:          informar("Tarea repetitiva")
4:          endif
5:      endif
```

3.5.3 Clasificar inválidos

Este método se refiere a la identificación de aquellos comandos que no se utilizan en tareas repetitivas (no hacen parte del diccionario de comandos) o los comandos que se utilizan pero son catalogados por el AFD como no aceptados, por no coincidir con una secuencia lógica. El método descrito se representa algorítmicamente de la forma:

Algoritmo 3. Algoritmo clasificador de comandos inválidos
Método **clasificar** (*comando*)

```
1:      if (comando in diccionario) then
2:          evaluarAFD (comando)
3:      else
4:          reciclar (comando,tipo_residuo)
5:      endif
```

El algoritmo comienza con la evaluación de los comandos en el diccionario, si hace parte del diccionario del AFD se ejecuta el método evaluarAFD, presentado en el Algoritmo 1, sobre dicho comando. En caso contrario, el comando pasa a ejecutar el método reciclar, que se presenta más adelante.[43]

3.5.4 Recopilar en papelera

Los comandos que son catalogadas como "inválidos" son enviados a la "papelera", el cual consiste de un método que define umbrales de acumulación y de tiempo, que al ser alcanzados reiniciará el AFD y el Agente, debido que el usuario no está realizando una tarea repetitiva.

Algoritmo 4. Algoritmo reciclador de comandos
Método **reciclar**(*comando*,*tipo*)

```
1:      if(tipo=tipo_residuo) then
2:      reciclaje=reciclaje+1
3:      if(reciclaje>=4000) or(tipo=tipo_negado)then
4:          tarea.reiniciar()
5:          actividad.reiniciar()
6:          AFD.reiniciar()
7:          reciclaje=0
8:      endif
```

El método recibe por parámetro los comandos que no pertenecen al diccionario, los negados por el autómata, o los tiempos en los cuales no se encuentran comandos, además recibe el tipo de comando. El método se encarga de contar todos los comandos recibidos, o la ausencia de comandos, los cuales llegan alrededor de cada 0,2 segundos. En caso de ser mayor o igual a 4000 llamadas al método que en tiempo serian alrededor de 15 minutos o recibir un comando de tipo negado; se toma la decisión de reiniciar la cola de tarea, la actividad presente, el estado de ejecución del AFD y el contador de solicitudes (reciclaje).

4 PRUEBAS PARA LA VALIDACIÓN DEUN ALGORITMO DE DETECCIÓN DE TAREAS REPETITIVAS

Para la validación del desarrollo propuesto se realizaron pruebas del algoritmo con dos casos artificiales y dos casos reales. Los casos artificiales equivalen a procesos

4Una actividad es un conjunto de comandos cuya finalidad es realizar una transacción u operación; por ejemplo, cortar y pegar un elemento, actualizar un elemento, eliminar, etc. [ver sección 2.1]

de copiado y pegado de elementos de software a otro, en el primer caso mediante una tarea repetitiva simple, y en el segundo, para una tarea con secuencia lógica. Los casos reales se basan en la migración masiva de datos de un software a otro, correspondientes a una empresa inmobiliaria.[44]

Tabla 4. Resultados de pruebas en primer caso artificial

Resultados caso N. 1				
Prueba	Actividades Reales	Actividades Detectadas	Duración (Seg.)	N. Comandos
1	4	3	34	28
2	4	3	35,5	33
3	5	3	30,8	29

4.1 *Caso artificial 1: Tareas repetitivas con operaciones fijas en sus actividades*

La tarea consiste en copiar los elementos de la primera fila del Software 1 a la primera columna del Software 2, siguiendo exactamente el orden mostrado en el software 2[Véase **¡Error! No se encuentra el origen de la referencia.**]. Para la ejecución de este caso se realizaron 3 pruebas, cuya respuesta se presenta a continuación.

En la

Resultados caso N. 1				
Prueba	Actividades Reales	Actividades Detectadas	Duración (Seg.)	N. Comandos
1	4	3	34	28
2	4	3	35,5	33
3	5	3	30,8	29

se presentan el contraste de los resultados arrojados por las tres pruebas realizadas. Los resultados arrojados muestran que fueron necesarias de cuatro a cinco actividades reales para que el algoritmo detectara tres actividades. De estas tres actividades detectadas el algoritmo encontró adecuadamente una tarea repetitiva con una duración promedio de 33,4 segundos invertidos en digitar de 28 a 33 comandos. Cabe aclarar que en la realización de las pruebas se imprimió una velocidad acorde a un digitador experto, lo cual dificultó la labor de una detección temprana de las actividades, requiriendo entre 4 y 5 ejecuciones para que se identificara la tarea repetitiva. Si bien los resultados en las pruebas son similares, la ligera diferencia en los tiempos se explica en las diferentes velocidades usadas en la ejecución de las actividades.

4.2 Caso real 1: Migración de datos manual entre sistemas de una empresa inmobiliaria.

El primer caso real de consiste en que una pequeña empresa Inmobiliaria utiliza a Microsoft Excel® como base de datos, almacenando los siguientes campos: referencia, fecha de alta, tipo de inmueble, tipo de operación, provincia, superficie, precio de venta, fecha de venta y vendedor. Esta pequeña empresa desea migrar los datos al software InmobiWebde tal forma que le permita realizar la operación diaria más rápidamente. El software InmobiWeb es una aplicación desarrollada en el entorno de prueba de la presente tesis de maestría, para garantizar la ejecución del experimento. [46] Dicho softwarees un sistema orientado a la web construido en lenguaje PHP®, framework CAKEPHP® y MySQL®. El sistema posee formularios que permiten la manipulación de las características de la información, tales como creación, actualización, eliminación, y

	A	B	C	D	E	F	G	H	I
1	Referencia	Fecha Alta	Tipo	Operación	Provincia	Superficie	Precio Vent	Fecha Vent	Vendedor
2	1	01/01/04	Parking	Alquiler	Lleida	291	2133903	19/06/04	Carmen
3	2	01/01/04	Local	Venta	Girona	199	1945424	19/04/04	Pedro
4	3	01/01/04	Oficina	Alquiler	Girona	82	712416	08/11/04	Joaquín
5	4	02/01/04	Parking	Alquiler	Girona	285	1815450	27/04/04	Jesús
6	5	02/01/04	Suelo	Venta	Tarragona	152	1138024	10/07/04	María
7	6	03/01/04	Industrial	Alquiler	Girona	131	953156	05/09/04	Pedro
8	7	03/01/04	Parking	Alquiler	Tarragona	69	406686	07/06/04	Pedro

consulta de los registros [véase **Figura 7**].

Figura 6. Muestra de base de datos de caso real 1
Fuente: (Superalumnos, 2007)

Figura 7. Muestra de software InmobiWeb para manejo de datos inmobiliarios

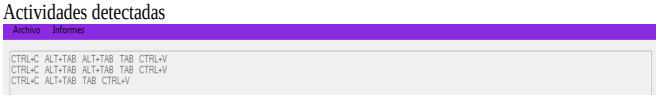
Ahora bien, el paso de la información desde Microsoft Excel® a InmobiWebse realiza de forma manual, tomando el valor de cada celda y llevándolo al correspondiente espacio en el sistema InmobiWeb.[47]

Para la evaluación de tareas repetitivas en este experimento se realizaron tres pruebas del caso real. A continuación se muestran los resultados arrojados por cada una de las pruebas, y se comparan los valores obtenidos.

Prueba 1:

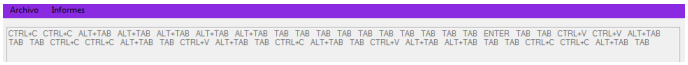
Comandos detectados

Actividades detectadas

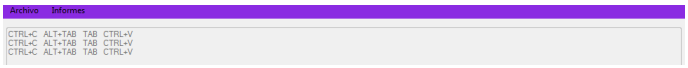


Prueba 2:

Comandos detectados

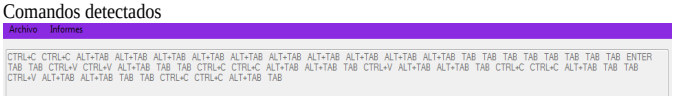


Actividades detectadas

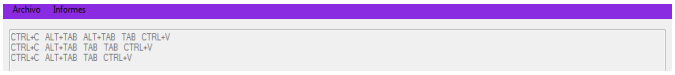


Prueba 3:

Comandos detectados



Actividades detectadas



En la **Tabla 5** se presentan el contraste de los resultados arrojados por las tres pruebas realizadas en el caso real 1. Los resultados muestran que tras la realización de cuatro actividades reales el algoritmo detectó tres de ellas. De las tres actividades detectadas el algoritmo encontró adecuadamente una tarea repetitiva con una duración promedio de 46,4 segundos invertidos en digitar de 37 a 51 comandos.[48]

Las diferencias en los valores de tiempos y comandos se explican en la velocidad empleada en la realización de cada prueba, donde se fue incrementando la velocidad en el desarrollo de las pruebas, lo cual condujo a la necesidad de digitar más comandos que garantizaran la detección.

Tabla 5. Resultados de pruebas en primer caso real

Resultados caso N. 3				
Prueba	Actividades Reales	Actividades Detectadas	Duración (Segundos)	N. Comandos
1	4	3	43,7	37
2	4	3	45,7	44
3	4	3	49,9	51

4.3 Discusión de resultados

Tras la realización de las pruebas con casos artificiales y reales se resaltan los siguientes hechos:

- En todos los casos analizados el algoritmo detecto de manera satisfactoria la realización de una tarea repetitiva mediante la ejecución de cinco o menos actividades reales.[53]
- El número de actividades necesarias para la identificación adecuada tiene una relación directa con la velocidad usada en la digitación.
- No se manifiestan diferencias entre la identificación de tareas repetitivas fijas y con secuencia lógica.
- El tiempo para la detección adecuada de la realización de una tarea repetitiva en todos los casos fue menor de un minuto.
- Existe una diferencia entre los comandos detectados en la ejecución de cada prueba, a pesar que en cada caso se realizó exactamente las mismas pruebas.[54]

En todos los casos se requirió realizar dos o más actividades adicionales reales antes que el algoritmo detectara efectivamente la tarea repetitiva.[55][56][57] El aumento en el tiempo y la necesidad de más comandos del teclado para realizar la detección está relacionada con la complejidad de la tarea repetitiva a detectar y a los posibles errores que el usuario puede tener mientras digita los comandos en el teclado.[58]

No obstante, otro factor que puede aumentar el tiempo y los comandos para realizar la tarea repetitiva son los errores de detección del Keylogger el cual detecta comandos cada 0.2 segundos y si los comandos son digitados a gran velocidad a menudo no se capturan.[59] La velocidad de escucha del Keylogger al aumentarse dificulta la detección de comandos que necesitan de varias teclas para formarse, debido a que detecta cada tecla de forma individual y no como parte de un comando, mientras que, si la velocidad se disminuye no se capturan teclas que sean digitadas rápidamente; por lo tanto, se decidió asumir las fallas que pudieran dar lugar tomando el tiempo de calibración del Keylogger en 0.2 segundos.[60][61]

5 CONCLUSIONES

En este trabajo se desarrolló satisfactoriamente un algoritmo novedoso para la detección de tareas repetitivas con secuencia lógica en sus actividades, basado en la clasificación de todos los comandos del teclado de Windows, y en la representación de los comandos, sus reglas y relaciones mediante el uso de Autómatas finitos deterministas.[62] El autómata, a su

vez, funciona como base de conocimiento para el Agente, quien lo utiliza como insumo en la construcción de las actividades que componen una tarea repetitiva.[63] El agente, así mismo, compara las actividades identificadas en la búsqueda de patrones que correspondan a una tarea repetitiva.

El excelente funcionamiento del algoritmo desarrollado se puede apreciar en la comparación de los resultados de las pruebas, con las cuales, se puede afirmar que,[64] el algoritmo cumple con su objetivo de detectar tareas repetitivas con secuencia lógica mediante el uso de comando del teclado. [65]

Cabe resaltar que, la representación de los comandos que componen la base de conocimientos se facilitó en gran medida por el uso adecuado de la estructura de autómatas, [66] mostrando que estos son la técnica de manejo de datos ideal para modelar una base de conocimientos sobre tareas repetitivas. [67]

Tras el cumplimiento del objetivo general de esta tesis de maestría, [68] la comunidad científica puede contar con una herramienta novedosa que permite la detección de diferentes tipos de tareas repetitivas realizadas a través de comandos del teclado. [69] [70] La detección facilita la posterior predicción y automatización de estas tareas tediosas para los usuarios abriendo el paso a que innumerables investigadores [71] y desarrolladores de software puedan trabajar en pro de suplir esta importante necesidad. [72][73]

6 REFERENCIAS BIBLIOGRÁFICAS

[1] Coronato, A., d'Acierno, A., & De Pietro, G. (2005). Automatic implementation of constraints in component based applications. Information and Software Technology , 47 (7), 497-509.

[2] Borràs, J., Moreno, A., & Valls, A. (2014). Intelligent tourism recommender systems: A survey. Expert Systems with Applications , 41 (16), 7370-7389

[3] Li, G., Lian, H., Feng, S., & Zhu, L. (2013). Automatic variable selection for longitudinal generalized linear models. Computational Statistics & Data Analysis , 61, 174-186.

[4] Alshalabi, H., Tiun, S., Omar, N., & Albared, M. (2013). Experiments on the Use of Feature Selection and Machine Learning Methods in Automatic Malay Text Categorization. Procedia Technology , 11, 748-754

[5] Joo, M., & Zhou, Y. (2008). A novel framework for automatic generation of fuzzy neural networks. Neurocomputing , 71 (4-6), 584-591.

[6] Chen, T., Zhang, X.-S., Guo, S.-Z., Li, H.-Y., & Wu, Y. (2013). State of the art: Dynamic symbolic execution for automated test generation. Future Generation Computer Systems , 29 (7), 1758-1773.

[7] Ruvini, J., & Dony, C. (2000). APE: Learning User's Habits to Automate Repetitive Tasks. Proceedings of the 2000 Conference on Intelligent User Interfaces.

[8] Kaber, D., & Prinzel, L. Adaptive and Adaptable Automation Design: A Critical Review of the Literature and Recommendations for Future Research. Hanover: NASA. 2006

[9] Sarter, N., Woods, D., & Billings, C. (1997). Automation Surprises. En G. Salvendry, Handbook of Human Factors & Ergonomics. Wiley.

[10] Parasuraman, R., Sheridan, T., & Wickens, C. (2000). A Model for Types and Levels of Human Interaction. IEEE Transactions on systems, man, and cybernetics - Part A: Systems and Humans , 286-297.

[11] Sagarna, R., Mendiburu, A., Inza, I., & Lozano, J. (2014). Assisting in search heuristics selection through multidimensional supervised classification: A case study on software testing. Information Sciences , 258, 122-139.

[12] Sheridan, T., & Verplank, W. (1987). Human and computer control of undersea teleoperators. Cambridge, MA: MIT Man-Machine Laboratory.

[13] Sah, M., & Wade, V. (2012). Automatic metadata mining from multilingual enterprise content. Web Semantics: Science, Services and Agents on the World Wide Web , 11, 41-62.

[14] Kuo, R., Huang, Y., Lin, C., Wu, Y., & Zulvia, F. (2014). Automatic kernel clustering with bee colony optimization algorithm. Information Sciences , 283 (1), 107-122.

[15] Les, T., Kruk, M., & Osowski, S. (2013). Automatic recognition of industrial tools using artificial intelligence approach. Expert Systems with Applications , 40 (12), 4777-4784.

[16]Flanagan, C. (2004). Automatic software model checking via constraint logic. Science of Computer Programming , 50 (1-3), 253-270.

[17]Gómez, A., Penadés, C., Canós, J., Borges, M., & Llavador, M. (2014). A framework for variable content document generation with multiple actors. Information and Software Technology , 56 (9), 1101-1121.

[18]Darragh, J., & Witten, I. (1991). Adaptive predictive text generation and the reactive keyboard. Interacting with Computers , 3 (1), 27-50.

[19]Osasan, K., & Stacey, T. (2014). Automatic prediction of time to failure of open pit mine slopes based on radar monitoring and inverse velocity method. International Journal of Mining Science and Technology , 24 (2), 275-280.

[20]Palomo-Duarte, M., García-Domínguez, A., & Medina-Bulo, I. (2014). Automatic dynamic generation of likely invariants for WS-BPEL compositions. Expert Systems with Applications , 41 (11), 5014-5055.

[21]Guo, Y., Wang, Y., & Liu, X. (2014). Real-time optical detection system for monitoring roller condition with automatic error compensation. Optics and Lasers in Engineering , 53, 69-78.

[22]Cypher, A. (1993). Watch what I do: Programming by demonstration. Cambridge, Mass.: MIT Press.

[23]Lieberman, H. (1993). Mondrian: A teachable graphical editor. En D. Cypher, In Watch what I do: Programming by demonstration. Cambridge, Mass.: MIT Press.

[24]Maes, P. (1994). Agents that reduce workand information overload. Communications of the ACM , 37 (7), 31-40.

[25]Maes, P., & Kozierok, R. (1993). Learning: Interface Agents. AAAI-93 Proceedings, (págs. 459-465).

[26]Holling, H., Bertling, J., & Zeuch, N. (2009). Automatic item generation of probability word problems. Studies in Educational Evaluation , 35 (2-3), 71-76.

[27]Myer, T. (2009). Apple Automator with AppleScript Bible. Indianapolis: Jhon Wiley & Sons.

[28]Mitchell, T., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experience with a learning personal assistant. Communications of the ACM , 37 (7), 81-91.

[29]Motoda, H. (1997). Machine learning techniques to make computers easier to use. Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97). San Francisco: Morgan Kaufmann.

[30]Caglayan, A., Snorrason, M., Jacoby, J., Mazzu, J., Jones, R., & Kumar, K. (1997). Learn sesame: a learning agent engine. Applied Artificial Intelligence , 11, 393-412.

[31]Armstrong, R., Freitag, D., Joachims, T., & Mitchell, T. (1995). WebWatcher: A learning apprentice for the World Wide Web. AAAI spring symposium on information gathering.

[32]. Debroy, V., & Wong, W. (2014). Combining mutation and fault localization for automated program debugging. Journal of Systems and Software , 90, 45-60.

[33]Ku, N., Jo, A., Ha, S., Rho, M., & Lee, K.-Y. (2012). Automatic generation of equations of motion for multibody system in discrete event simulation framework. Procedia Technology , 1, 55-64.

[34]Guevara, C. (2012). Reconocimiento de patrones para identificación de usuarios en accesos informáticos. Madrid, España: Universidad Complutense de Madrid, Tesis de Maestría.

[35]Argall, B., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. Robot. Auton. Syst. , 469-483.

[36]Christl, A., Koschke, R., & Storey, M. (2007). Automated clustering to support the reflexion method. Information and Software Technology , 49 (3), 255-274.

[37]Yigit, T., Isik, A., & Ince, M. (2014). Multi Criteria Decision Making System for Learning Object Repository. Procedia - Social and Behavioral Sciences , 141, 813-816.

[38]Zhan, Y., & Clark, J. (2008). A search-based framework for automatic testing of MATLAB/Simulink models. Journal of Systems and Software , 81 (2), 262-285.

[39]Derrode, S., & Pieczynski, W. (2013). Unsupervised data classification using pairwise Markov chains with automatic copulas selection. Computational Statistics & Data Analysis , 63, 81-98.

[40]Hopcroft, J., Motwani, R., & Ullman, J. (2001). Introduction to Automata Theory, Languages, and Computation. Massachusetts, USA: Addison-Wesley.

[41]Liu, D., Cui, B., Liu, Y., & Zhong, D. (2013). Automatic control and real-time monitoring system for earth–rock dam material truck watering. Automation in Construction , 30, 70-80.

[42]Pérez, B., & Polo, M. (2009). Generación automática de casos de prueba para Líneas de Producto de Software. Revista Española de Innovación, Calidad e Ingeniería del Software , 5 (2), 17-27.

[43]Dunn, k. (2004). Automatic update risks: can patching let a hacker in? Network Security , 2004 (7), 5-8.

[44]Farjoodi, J., & Soroushian, A. (2001). Efficient Automatic Selection of Tolerances in Nonlinear Dynamic Analysis. En A. Zingoni, Structural Engineering, Mechanics and Computation (págs. 853-859). Oxford: Elsevier Science.

[45]Fernández, A., Gómez, A., Lecumberry, F., Pardo, A., & Ramírez, I. (2014). Pattern Recognition in Latin America in the “Big Data” Era. Pattern Recognition , In press.

[46]Yajun, Z., & Qian, Q. (2012). A New Type of Automatic Monitoring System of Static and Dynamic Displacement on Dam and Slope. Procedia Engineering , 43, 387-392.

[47]Zhang, Y., Dang, Y., Chen, H., Thurmond, M., & Larson, C. (2009). Automatic online news monitoring and classification for syndromic surveillance. Decision Support Systems , 47 (4), 508-517.

[48]Zhang, Y., Li, Y., & Zheng, W. (2013). Automatic software deployment using user-level virtualization for cloud-computing. Future Generation Computer Systems , 29 (1), 323-329.

[49]Superalumnos. (07 de 11 de 2007). Superalumnos.net. Recuperado el 11 de 08 de 2014, de Base de datos de ejemplo: Inmobiliaria: http://superalumnos.net/base-de-datos-de-ejemplo-inmobiliaria

[50]Sutton, R., & Barto, A. (1998). Reinforcement Learning: An Introduction. Cambridge: Cambridge University Press.

[51]Van Royen, K., Poels, K., Daelema, W., & Vandebosch, H. (2014). Kathleen Van Royen, Karolien Poels, Walter Daelemans, Heidi Vandebosch, Automatic monitoring of cyberbullying on social networking sites: From technological feasibility to desirability. Telematics and Informatics , in Press.

[52]Zhou, P., Li, D., Wu, H., & Cheng, F. (2011). The automatic model selection and variable kernel width for RBF neural networks. Neurocomputing , 74 (17), 3628-3637.

[53]Arcuri, A., & Yao, X. (2008). Search based software testing of object-oriented container. Information Sciences , 178 (15), 3075-3095.

[54]Xu, J. (2012). Rule-based automatic software performance diagnosis and improvement. Performance Evaluation , 69 (11), 525-550.

[55]Zhang, Y., Zhang, L., & Alamgir, M. (2014). Adaptive 3D Facial Action Intensity Estimation and Emotion Recognition. Expert Systems with Applications , In press.

[56]Alonso, D., Pastor, J., Sánchez, P., Álvarez, B., & Vicente-Chicote, C. (2012). Generación Automática de Software para Sistemas de Tiempo Real: Un Enfoque basado en Componentes, Modelos y Frameworks. Revista Iberoamericana de Automática e Informática Industrial RIAI , 9 (2), 170-181.

[57]Wang, Y., Wang, D., & Fang, W. (2014). Automatic node selection and target tracking in wireless camera sensor networks. Computers & Electrical Engineering , 40 (2), 484-493.

[58]Zienkiewicz, O., Taylor, R., & Zhu, J. (2013). Automatic Mesh Generation. En O. Zienkiewicz, R. Taylor, & J. Zhu, The Finite Element Method: its Basis and Fundamentals (Septima edición ed., págs. 573-640). Oxford: Butterworth-Heinemann.

[59]Mokhtarian, F., & Abbasi, S. (2005). Robust automatic selection of optimal views in multi-view free-form object recognition. Pattern Recognition , 38 (7), 1021-1031.

[60]Moral, S. (2006). Modelos de Computación I. Granada, España: Guias de asignatura. Universidad de Granada.

[61]Seok, J., & Seong, H. (2015). Automatic generation algorithm of expected results for testing of component-based software system. Information and Software Technology , 57, 1-20.

[62]Shih, H. (2014). A robust occupancy detection and tracking algorithm for the automatic monitoring and commissioning of a building. Energy and Buildings , 77, 270-280.

[63]Varela-Vaca, A., & Gasca, R. (2013). Towards the automatic and optimal selection of risk treatments for business processes using a constraint programming approach. Information and Software Technology , 55 (11), 1948-1973.

[64]Wang, Q., & Yu, X. (2014). Ontology based automatic feature recognition framework. Computers in Industry , 65 (7), 1041-1052.

[65]Shahriar, H., & Zulkernine, M. (2011). Taxonomy and classification of automatic monitoring of program security vulnerability exploitations. Journal of Systems and Software , 84 (2), 250-269.

[66]Arcuri, A. (2011). Evolutionary repair of faulty software. Applied Soft Computing , 11 (4), 3494-3514.

[67]Cobo, L., Subramanian, K., Isbell, C., Lanterman, A., & Thomaz, A. (2014). Abstraction from demonstration for efficient reinforcement learning in high-dimensional domains. Artificial Intelligence , 103-128.

[68]Dehua, W., Pan, L., Bo, L., & Zeng, G. (2012). Water Quality Automatic Monitoring System Based on GPRS Data Communications. Procedia Engineering , 28, 840-843.

[69]Dominguez, A., Tojo, J., & Castier, M. (2002). Automatic implementation of thermodynamic models for reliable parameter estimation using computer algebra. Computers & Chemical Engineering , 26 (10), 1473-1479.

[70]Liu, D., Yang, Z., Tang, C., Wang, J., & Liu, Y. (2004). An automatic monitoring system for the shiplock slope of Wuqiangxi Hydropower Station. Engineering Geology , 76 (1-2), 79-91.

[71]Lu, C., & Lu, Z. (2008). Local feature extraction for iris recognition with automatic scale selection. Image and Vision Computing , 26 (7), 935-940.

[72]Lucey, P., Cohn, J., Prkachin, K., Solomon, P., Chew, S., & Matthews, I. (2012). Painful monitoring: Automatic pain monitoring using the UNBC-McMaster shoulder pain expression archive database. Image and Vision Computing , 30 (3), 197-205.

[73]Messelis, T., & De Causmaecker, P. (2014). An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem. European Journal of Operational Research , 233 (3), 511-528.