Gestión de la Tecnología - Parcial 1 - Coherencia, Cohesión, trama, propósito y paratextos.

Responda las siguientes preguntas antes de comenzar. Por favor, lea detenidamente cada instrucción para realizar el examen correctamente. Good luck!

Puntos: 37/50

1

Dirección de correo electrónico *

atrejovallejos@alumno.unlam.edu.ar

2

Indicar APELLIDO , NOMBRE/S (todo en mayúscula y separado por una coma). *

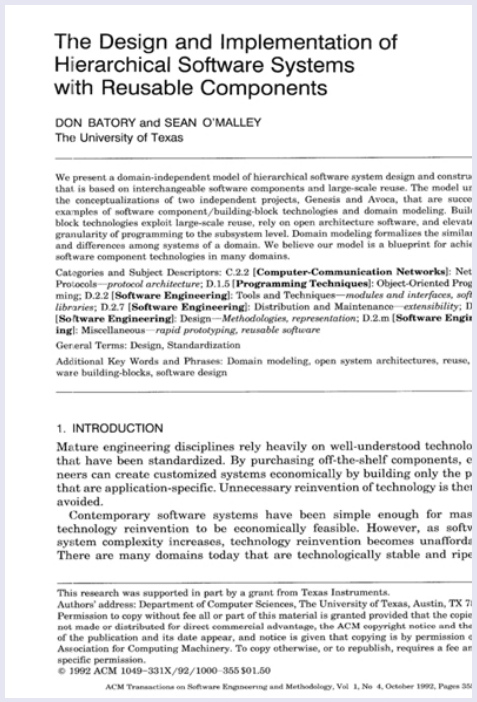
TREJO VALLEJOS, ARIEL ERNESTO

3

¿Cuál es su D.N.I? (usar puntos, por ejemplo: 00.000.000) *

28.323.371

Observe el siguiente texto y elija la opción correcta.



The Design and Implementation of Hierarchical Software Systems with Reusable Components

DON BATORY and SEAN O'MALLEY
The University of Texas

We present a domain-independent model of hierarchical software system design and constru
that is based on interchangeable software components and large-scale reuse. The model ur
the conceptualizations of two independent projects, Genesis and Avoca, that are succe
examples of software component/building-block technologies and domain modeling. Buil
block technologies exploit large-scale reuse, rely on open architecture software, and elevat
granularity of programming to the subsystem level. Domain modeling formalizes the simila
and differences among systems of a domain. We believe our model is a blueprint for achi
software component technologies in many domains.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Net
Protocols—*protocol architecture*; D.1.5 [**Programming Techniques**]: Object-Oriented Prog
ming; D.2.2 [**Software Engineering**]: Tools and Techniques—*modules and interfaces, soft
libraries*; D.2.7 [**Software Engineering**]: Distribution and Maintenance—*extensibility*; D
[**Software Engineering**]: Design—*Methodologies, representation*; D.2.m [**Software Engi
ing**]: Miscellaneous—*rapid prototyping, reusable software*

General Terms: Design, Standardization

Additional Key Words and Phrases: Domain modeling, open system architectures, reuse,
ware building-blocks, software design

1. INTRODUCTION

Mature engineering disciplines rely heavily on well-understood technolo
that have been standardized. By purchasing off-the-shelf components, e
neers can create customized systems economically by building only the p
that are application-specific. Unnecessary reinvention of technology is the
avoided.

Contemporary software systems have been simple enough for mas
technology reinvention to be economically feasible. However, as softw
system complexity increases, technology reinvention becomes unaffordz
There are many domains today that are technologically stable and ripe

This research was supported in part by a grant from Texas Instruments.
Authors' address: Department of Computer Sciences, The University of Texas, Austin, TX 7

4

Este texto es un/una... *
(5 puntos)

- [ ] Capítulo de un libro
- [ ] Revista
- [ ] Índice o manual.
- [x] Artículo científico (paper).

5

Función o propósito comunicativo: *
(5 puntos)

- ( ) Apelar
- (•) Informar
- ( ) Persuadir

6

¿Cual es la trama u organización textual que **predomina**? *
(5 puntos)

- ( ) Descriptivo
- (•) Argumentativo

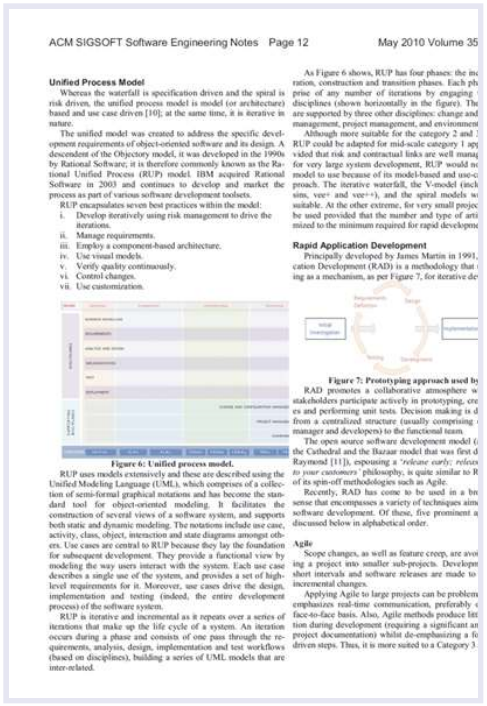
7

Si los hay, marque los paratextos **VERBALES** que puede encontrar (*puede marcar mas de una opción*): *
(6 puntos)

- [x] Cambio de tamaño de letra
- [x] Negritas
- [ ] Viñetas
- [x] Fuente del texto
- [ ] Número de pagina
- [x] Pie de pagina

- [x] Titulo
- [x] Cursivas
- [ ] Columnas
- [x] Subtitulo

Observe el siguiente texto y elija la opción correcta.



ACM SIGSOFT Software Engineering Notes Page 12 May 2010 Volume 35

Unified Process Model

Whereas the waterfall is specification driven and the spiral is risk driven, the unified process model is model (or architecture) based and use case driven [10]; at the same time, it is iterative in nature.

The unified model was created to address the specific development requirements of object-oriented software and its design. A descendent of the Objectory model, it was developed in the 1990s by Rational Software; it is therefore commonly known as the Rational Unified Process (RUP) model. IBM acquired Rational Software in 2003 and continues to develop and market the process as part of various software development toolsets.

RUP encapsulates seven best practices within the model:

i. Develop iteratively using risk management to drive the iterations.
ii. Manage requirements.
iii. Employ a component-based architecture.
iv. Use visual models.
v. Verify quality continuously.
vi. Control changes.
vii. Use customization.

Figure 6: Unified process model.

RUP uses models extensively and these are described using the Unified Modeling Language (UML), which comprises of a collection of semi-formal graphical notations and has become the standard tool for object-oriented modeling. It facilitates the construction of several views of a software system, and supports both static and dynamic modeling. The notations include use case, activity, class, object, interaction and state diagrams amongst others. Use cases are central to RUP because they lay the foundation for subsequent development. They provide a functional view by modeling the way users interact with the system. Each use case describes a single use of the system, and provides a set of high-level requirements for it. Moreover, use cases drive the design, implementation and testing (indeed, the entire development process) of the software system.

RUP is iterative and incremental as it repeats over a series of iterations that make up the life cycle of a system. An iteration occurs during a phase and consists of one pass through the requirements, analysis, design, implementation and test workflows (based on disciplines), building a series of UML models that are inter-related.

As Figure 6 shows, RUP has four phases: the inc
ration, construction and transition phases. Each ph
prise of any number of iterations by engaging
disciplines (shown horizontally in the figure). The
are supported by three other disciplines: change and
management, project management, and environment

Although more suitable for the category 2 and
RUP could be adapted for mid-scale category 1 ap
vided that risk and contractual links are well manag
for very large system development, RUP would n
model to use because of its model-based and use-c
proach. The iterative waterfall, the V-model (incl
sins, vee+ and vee++), and the spiral models w
suitable. At the other extreme, for very small projec
be used provided that the number and type of arti
mized to the minimum required for rapid developme

Rapid Application Development

Principally developed by James Martin in 1991,
cation Development (RAD) is a methodology that
ing as a mechanism, as per Figure 7, for iterative de

Figure 7: Prototyping approach used by

RAD promotes a collaborative atmosphere w
stakeholders participate actively in prototyping, cre
es and performing unit tests. Decision making is d
from a centralized structure (usually comprising
manager and developers) to the functional team.

The open source software development model (
the Cathedral and the Bazaar model that was first d
Raymond [11]), espousing a *"release early; releas
to your customers"* philosophy, is quite similar to R
of its spin-off methodologies such as Agile.

Recently, RAD has come to be used in a br
sense that encompasses a variety of techniques atm
software development. Of these, five prominent a
discussed below in alphabetical order.

Agile

Scope changes, as well as feature creep, are avoi
ing a project into smaller sub-projects. Developm
short intervals and software releases are made to
incremental changes.

Applying Agile to large projects can be problem
emphasizes real-time communication, preferably
face-to-face basis. Also, Agile methods produce lit
tion during development (requiring a significant am
project documentation) whilst de-emphasizing a fo
driven steps. Thus, it is more suited to a Category 3

8

Este texto es un/una... *
(5 puntos)

- ( ) Revista
- (•) Artículo científico (paper)
- ( ) Capítulo de un libro
- ( ) Índice o manual.

9

Función o propósito comunicativo: *
(5 puntos)

- Apelar
- Informar
- Persuadir

10

¿Cual es la trama u organización textual que **predomina**? *
(5 puntos)

- Descriptivo
- Argumentativo

11

Si los hay, marque los paratextos **VERBALES** que puede encontrar (*puede marcar mas de una opción*): *
(5 puntos)

- [x] Titulo
- [x] Columnas
- [x] Cambio de fuente
- [x] Viñetas
- [x] Subtitulo
- [ ] Pie de pagina

- [ ] Número de pagina
- [ ] Negritas
- [x] Cursivas

Lea el siguiente extracto de texto y responda:

"Software is the stuff that makes your computer do things for **you**. The computer without software would be like a home entertainment system with no tapes, CD's, or movies - you have the machine, but there's nothing to play on **it**. Software is continually developed. Each time the software maker (Microsoft, Adobe, Corel, etc) develops a new version of their software they assign it a version number. Before Microsoft Word 7, there was Microsoft Word 6.0.1, and before that Word 6.0. The larger the developments made to the software, the larger the version number changes. Usually a large change will result in a whole number upgrade; a small change may
result in a tenth of a decimal place."

12

La palabra "**you**" **subrayada y en negrita** en el texto refiere a: *
(3 puntos)

- ( ) Software
- ( ) Computer
- (•) The reader

13

La palabra "**it**" **subrayada y en negrita** en el texto refiere a: *
(3 puntos)

- Home entertainment system
- Software
- Movies
- Computer

14

Lea el texto nuevamente e indique cuáles de las siguientes palabras extraídas son palabras **estructurales** (*más de una opción es posible*): *
(3 puntos)

- [x] The
- [x] On
- [ ] Adobe
- [ ] Software
- [ ] Play
- [x] Like
- [ ] Version