3C TIC *(Edición núm. 8)* Vol.3 – Nº 1 | Marzo – Junio 2014, 36 - 51
*Área de Innovación y Desarrollo, S.L.*
**ISSN: 2254 – 6529**

Recepción: 09 de enero de 2014
Aceptación: 11 de febrero de 2014
Publicación: 27 de marzo de 2014

# ALEATORIEDAD EN LA EVALUACIÓN DE LOS MOOC

## RANDOMNESS IN MOOC ASSESSMENT

Enrique Sánchez Acosta[1]

Juan José Escribano Otero[2]

1. Ingeniero de Software en Soft S.A. Doctorando en Tecnologías de Información Aplicadas. Departamento de informática, automática y comunicaciones, Universidad Europea. E-mail: esacosta@gmail.com
2. Director del departamento de informática, automática y comunicaciones de la Universidad Europea. E-mail: juanjose.escribano@uem.es

3C TIC (Edición núm. 8) Vol.3 – Nº 1 | Marzo – Junio 2014, 36 - 51
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

## RESUMEN

El hombre siempre se ha visto afectado por la aleatoriedad de algunos fenómenos y ha sentido la necesidad de generarla para diversos experimentos científicos, tanto en el mundo de las matemáticas como en el de las ciencias de la computación, hoy ligadas más que nunca a los nuevos paradigmas educativos.

El auge que están experimentando los cursos masivos online (MOOC) en los últimos años, está haciendo que cada vez sea más necesario analizar la fiabilidad de los test de evaluación que aparecen en las distintas plataformas de aprendizaje online. No solamente en cuanto al funcionamiento de estas plataformas, sino también para equilibrar las preguntas y respuestas de los exámenes tipo test. No debe olvidarse que ante un curso con miles de estudiantes de todo el mundo, algunos de ellos tratarán de hacer trampas para obtener los mejores resultados en el menor tiempo posible.

## ABSTRACT

Mankind has always been impressed by the randomness of some events and felt the need to generate it for various scientific experiments, both in the world of mathematics as the science of computing, today more than ever linked to new educational paradigms.

The boom is experimenting the massive online courses (MOOC) in recent years, is becoming increasingly necessary to examine the reliability of the assessment test that appear on different platforms of online learning. Not just in terms of the performance of these programs, but also to balance the questions and answers of multiple choice exams. Do not forget that before a course with thousands of students around the world, some of them try to cheat to get the best results in the shortest time possible.

## PALABRAS CLAVE

MOOC, aleatorio, test, congruencial, e-learning.

## KEYWORDS

MOOC, random, test, congruential, e-learning

# INTRODUCCIÓN

El estudio realizado para este artículo no trata de perder mucho tiempo en analizar los test de evaluación del curso para comprobar su fiabilidad con métodos estadísticos, que aunque a priori parecerían la opción más válida, se complican demasiado con la gran cantidad de opciones existentes en los nuevas plataformas educativas online, donde los antiguos test de opción múltiple están adoptando nuevas formas de evaluación automática en las plataformas que soportan los cursos masivos online. En vez de ello, se va a tener una batería de pruebas que nos indique, con aleatoriedad, si alguien pudiera superar el curso, y servirá también para comprobar que la plataforma responde correctamente a nuestras expectativas. Para ello, no sólo es necesario saber que se puede realizar este tipo de pruebas, sino conocer cómo funcionan y cómo se puede sacar el mayor provecho de los números aleatorios y pseudo-aleatorios que pueden utilizarse en la creación de estas herramientas.

Desde sus inicios se han utilizado para innumerables aplicaciones, juegos, criptografía, estudios científicos, estudios estocásticos del modelo de Montecarlo, etc. No debe extrañar que comiencen a ser utilizados en la educación a distancia, basada en los nuevos avances tecnológicos en computación.

Uno de los principales problemas en la generación de números aleatorios es su calidad, de ésta depende la fiabilidad de numerosas aplicaciones, para ello deben basarse en los modelos matemáticos y en la potencia de las computadoras.

El estudio del azar comenzó realmente mucho antes del siglo XVII, sin embargo fue en este siglo cuando varios matemáticos se interesaron por la resolución de los juegos establecidos como juegos de azar para la época, fue entonces cuando nació la probabilidad. Blaise Pascal, incitado por Antoine Gombauld, resolvió el problema de que saliesen dos seises en dos dados en veinticuatro tiradas (Fernández, 2006), y a partir de ese momento la teoría de la probabilidad no ha parado de formar parte de las matemáticas hasta nuestros días. Sin embargo, faltaba poder generar esa aleatoriedad para que pudiera demostrar los problemas probabilísticos.

Fue John Von Newman el que dijo por los años cuarenta “el computador abrirá un nuevo enfoque para la estadística matemática, y el cálculo de experimentos…” (Herrera, 2000). En esta época varios matemáticos comenzaron a simular el método de Montecarlo, y necesitaron de la generación de estos números aleatorios. Sin embargo fue D. H. Lehmer quien propuso el generador lineal de congruencia como el primer método de generación de números aleatorios (Lehmer, 1951), antes de esto la generación de números aleatorios se realizaba únicamente a través de elementos físicos, como los dispositivos de disco giratorio iluminado de Babington – Smith y Kendall (Herrera, 2000).

## NÚMERO ALEATORIO Y PSEUDO-ALEATORIO

Un número aleatorio no es más que áquel que puede ser generado a partir de fuentes de aleatoriedad como dispositivos físicos, dados, ruletas, etc., donde no se puede controlar todos los aspectos posibles que pueden influir en su generación. Por ejemplo, la generación de los rayos, o su frecuencia en el tiempo, métodos atmosféricos, etc.

Un número pseudo-aleatorio, sin embargo, es aquel que, aún pareciéndose a los números aleatorios, siguen un patrón de naturaleza matemática. Son una simple emulación de un número aleatorio, sin embargo serán éstos los más utilizados para su uso en experimentos científicos, juegos de azar, etc. (Bu, 1993)

3C TIC (Edición núm. 8) Vol.3 – Nº 1 | Marzo – Junio 2014, 36 - 51
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

# NECESIDAD DEL ESTUDIO

## PREGUNTAS DE RESPUESTA MÚLTIPLE

A través del estudio sobre las tasas de finalización de los MOOC llevado a cabo por la doctora Kathy Jordan, y que puede verse en [http://www.katyjordan.com/MOOCproject.html], es posible observar que la gran mayoría de los MOOC analizados utilizan preguntas tipo test, sobretodo de opción múltiple. Este tipo de test de evaluación consta de una proposición que constituye el enunciado de la cuestión y una serie de respuestas que pueden adoptar diferentes formas (palabras, frases, símbolos, números, etc.). Esta prueba está considerada como la más objetiva de las existentes y según la metodología utilizada en el entorno virtual de aprendizaje "eValpa", diseñado para desarrollar el Programa Semipresencial de Formación Continua en Educación Médica «Valpa 2011», hay que tener en cuenta una serie de aspectos para realizar bien este tipo de examen.

**Enunciado**

Los enunciados tienen que ser categóricos, de tal modo que sólo admitan una respuesta correcta.

- Cada pregunta debe contener tan solo una idea.
- Hay que evitar plantear dos preguntas en el mismo ítem.
- Las preguntas se plantean afirmativamente.
- Evitar en lo posible las negaciones, de aparecer en el cuerpo de la pregunta subrayarla para llamar la atención sobre ella.
- Hay que evitar que la formulación de una pregunta resuelva otra pregunta anterior.
- Si se pretende evaluar el conocimiento de términos, conceptos, etc., éstos deben figurar en las preguntas, mientras que las descripciones o definiciones figuran en las respuestas alternativas.

**Respuestas**

Hay que evitar:

- El uso de palabras como "siempre, nunca, etc." que evocan respuestas correctas.
- La utilización de palabras que den pistas al alumno.
- Dar indicios que puedan sugerir la respuesta correcta.
- Cuidar que los distractores sean homogéneos, es decir semejantes en el contenido y en el número total de palabras.
- Las frases largas y complejas tienden a ser correctas. Conviene dar a cada enunciado una extensión similar. El texto del distractor debe ser breve e inteligible.

- Todas las respuestas alternativas, correctas e incorrectas deben tener cierta homogeneidad. Si los distractores actúan llamando la atención por su falta de sentido común la prueba pierde objetividad.
- El contenido del distractor debe entenderse sin necesidad de leer las respuestas alternativas.
- Utilizar distractores plausibles y lógicos, cada uno por su contenido y naturaleza debe aparentar tener relación con la pregunta.
- No deben emplearse las mismas palabras o frases que figuren en los libros de texto, salvo que se trate de memorizaciones necesarias.
- Evitar negaciones reiteradas, tras unos cuantos "no'', ''nunca', etc., nadie puede comprender de qué va la frase.
- No abusar de las frases como "ninguna de las respuestas anteriores" como distractor o respuesta correcta.

El principal inconveniente de estas pruebas, aparte del margen de azar, es que no se valoran adecuadamente el pensamiento creador y la dificultad de construir los ítems.

Este margen de azar es el que se ha tenido en cuenta a la hora de desarrollar este artículo, dado que resulta muy complicado evaluar la probabilidad de todas las combinaciones que pueden presentarse dentro de un MOOC, no sólo al hablar de preguntas de respuesta múltiple, sino también cuando éstas se mezclan con otro tipo de evaluaciones dentro de un mismo examen.

## BATERÍA DE TEST

En la siguiente imagen (ver Ilustración 1) puede observarse como en muchas ocasiones es la aleatoriedad la que nos dicta la respuesta correcta en un examen. Si se utiliza un examen tipo test en un entorno virtual de aprendizaje para un curso masivo online de 4 respuestas donde 1 de ellas es la correcta, cada error debería descontar 1/3, cosa que no se está aplicando en la mayoría de los cursos.

El motivo de esta puntuación negativa es para que el alumno que utiliza los sistemas aleatorios saque una nota de 0 en esas preguntas que no sabe. Esto se obtiene gracias a la distribución uniforme, la esperanza matemática y la independencia estadística. De modo que:

Si la nota final es el sumatorio de todas las notas:

$$\text{nota} = \sum_{i=1}^{N} n_i$$

$$n_i = \{1, \text{probabilidad}\ \tfrac{1}{M} \quad -X, \text{probabilidad}\ 1 - \tfrac{1}{M}\}$$

Donde *M* es el número de respuestas posibles (4)

$$E[n_i] = \frac{1}{M} * 1 + (1 - \frac{1}{M}) * (-X) = 1/M[1 - X(M - 1)$$

Si se quiere obtener E[nota] = 0, despejando se obtiene *X* que es el valor a restar por cada pregunta.

$$X = \frac{1}{M - 1}$$

Sin embargo, no basta con esto para determinar que todo funciona correctamente, porque resulta muy complejo el cálculo cuando hay muchos test y cuando no todos ellos tienen el mismo número de preguntas, no solo entre los test sino entre las mismas preguntas con diferentes opciones. ("Ciencia explicada: Exámenes tipo test, esperanza matemática y porqué los errores DEBEN restar," n.d.)

3C TIC (Edición núm. 8) Vol.3 – Nº 1 | Marzo – Junio 2014, 36 - 51
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

# MARCO TEÓRICO

## ALGORITMOS DE GENERACIÓN DE NÚMEROS ALEATORIOS

### Generador congruencial lineal de Lehmer

Esta es la base de prácticamente todos los algoritmos utilizados actualmente para generar números aleatorios, con diversas variaciones. El sistema es relativamente sencillo, la secuencia de números aleatorios se obtiene en base a la siguiente fórmula:

$$X_{f+1} = (aX_j + c) \bmod m$$

$X_0$ es el valor de la semilla inicial, *a* es el multiplicador, *c* el incremento y *m* el módulo.

Por ejemplo, tomando como valores, $m = 2^5 = 32, a = 5, x0 = 1 \text{ y } c = 3$ se obtiene la siguiente secuencia de números, que tiene longitud máxima:

1-8-11-26-5-28-15-14-9-16-19-2-13-4-23-22-17-24-27-10-21-12-31-30-25-0-3-18-29-20-7-6-1

Si se profundiza más en esta fórmula se observa que los números aleatorios degeneran y comienzan a repetirse a partir de ciertas secuencias, por lo tanto no es muy bueno generar los números aleatorios con datos completamente al azar, sino determinar una serie de números mágicos que nos consigan una serie aleatoria suficiente para nuestro experimento. Es por ello que hoy en día siguen apareciendo nuevos métodos o variantes de Lehmer para generar mejores secuencias de números aleatorios. (Herrera, 2000)

### Generador Shift-Register

$$X_{n+k} = \left(\sum_{i=0}^{k-1} a_i X_{n+i}\right) (\bmod 2)$$

En este caso las *X* y las *a* son 0 o 1. Existen dos formas para este algoritmo. La primera utiliza los 0 y 1 como bits para generar una secuencia de bits que corresponden a un número binario y la segunda forma es a partir de un número dado, lo modifica y genera otro aleatorio cambiando los 0 y 1 del número binario.

### Generador de Lagged-Fibonnaci

Con el crecimiento de los computadores con procesadores paralelos, este algoritmo ha tomado mucho más auge. Su nombre viene definido porque se parece al uso de la serie de Fibonacci. Utiliza el algoritmo de Lehmer mejorándolo en la posibilidad de crear operaciones independientes que pueden ser calculadas en paralelo por los computadores actuales.

Una de las ventajas de este generador es que puede utilizar números de coma flotante, muy usados por los computadores para la optimización de cálculos de números decimales. De este modo, se evita la transformación en enteros que utilizaban los métodos de Lehmer y que para un computador son operaciones costosas en tiempo de procesamiento.

$$X_n = (X_{n-j} \blacksquare X_{n-k}) \bmod (M), j < k, M = 2^m$$

Donde ■ es cualquier operador (+,-,*, etc.).

El problema de este algoritmo radica en que es necesaria una tabla con los primeros *k* elementos. Estos son los que se van actualizando de manera circular. (Wikipedia, 2013)

## Método de Knuth

Quizá uno de los métodos más utilizados hoy en día. Se basa en el generador congruencial lineal de Lehmer, por lo que aqueja también del problema de la selección de la semilla.

$$u_t = \frac{(aZ_{t-1} + c) \bmod m}{m}$$

La calidad, como en todos los sistemas derivados del generador congruencial de Lehmer depende de los valores elegidos de $Z_0$, *a*, *c* y *m*. Knuth estableció una serie de criterios a seguir para elegir estos valores y tener éxito con ellos:

***Selección de la semilla***

Debe ser un entero entre 1 y m-1. Aunque hoy en día suele utilizarse la fórmula:

$$Z_0 = \text{Abs}(Z_0) \bmod m$$

***Selección del módulo***

El módulo debe ser muy grande, suele utilizarse la longitud de palabra del computador, como en el ejemplo de Lehmer. (“Simulación de sistemas,” n.d.)

*Método de Knuth* ($Z_0$)
*Inicio*

$Z_0 \leftarrow Z_0 \bmod m$

$Z \leftarrow \text{Anterior}_Z(Z_0)$

$u \leftarrow \frac{z}{m}$

devolver u

*Fin*

## ÚLTIMOS AVANCES EN GENERACIÓN DE ALGORITMOS ALEATORIOS

En los últimos años se ha avanzado mucho en la generación de números aleatorios dada su importancia en el campo de la criptografía, seguridad y en los juegos de azar o de suma 0 como el ajedrez, gracias a los nuevos avances en las tablas hash de Zobrist (Zobrist, 1970).

### Mersenne twister

Se trata de un generador desarrollado en 1997 por Matsumoto y Nishimura. Su generación tiene que ver con los números primos de Mersenne y dependiendo de los números primos utilizados en su generación existen diversos métodos, tanto para 32 bits como para 64 bits.

La eficacia de este algoritmo radica en que con una palabra de k bits es capaz de generar al menos una distribución uniforme de 0 a $2^k$ números.

Existen muchas implementaciones para este algoritmo y aunque resulta mucho más complejo que los vistos con anterioridad no cabe duda que es mucho más eficiente que el resto.

### Random KISS

Es un sistema creado muy recientemente por George Marsaglia y cuenta con ciertas ventajas sobre el de Mersenne citado anteriormente. (Marsaglia, 1968)

Es cuatro veces más rápido que Mersenne, e independiente de la plataforma, funciona perfectamente en máquinas de 64 bits, y permite el procesamiento en paralelo en los actuales computadores, su implementación resulta muy sencilla.

Un pequeño ejemplo en lenguaje C++ de un motor de ajedrez llamado AlfilChess [ www.alfilchess.es] y que se utilizó en este estudio para comprobar la fiabilidad y rendimiento de este algoritmo es el siguiente:

```
static unsigned long long
x=1234567890987654321ULL,c=1234561234561234567ULL,
y=362436362436362436ULL,z=1066149217761810ULL,t;

#define MWC (t=(x<<5+c), c=(x>>6), x+=t, c+=(x<t), x)
#define XSH ( y^=(y<<13), y^=(y>>17), y^=(y<<43) )
#define CNG ( z=6906969069LL*z+1234567 )
#define KISS (MWC+XSH+CNG)

class cRandomKiss
{
public:
  cRandomKiss() { raninit(); }
  struct S { uint64 a, b, c, d; } s;

  uint64 rotate(uint64 x, uint64 k) const
  {
    return (x << k) | (x >> (64 - k));
  }

  //-- Devuelve un entero de 64 bits [0, 2^64 - 1]
  uint64 rand64()
  {
```

```
  const uint64 e = s.a - rotate(s.b,  7);
  s.a = s.b ^ rotate(s.c, 13);
  s.b = s.c + rotate(s.d, 37);
  s.c = s.d + e;
  return s.d = e + s.a;
 }

 //-- Inicializar la semilla
 void raninit()
 {
  s.a = 0xf1ea5eed;
  s.b = s.c = s.d = 0xd4e12c77;
 }
};

extern cRandomKiss m_RKiss;
```

3C TIC *(Edición núm. 8)* Vol.3 – Nº 1 | Marzo – Junio 2014, 36 - 51
Área de Innovación y Desarrollo, S.L.
**ISSN: 2254 – 6529**

## MÉTODO

Evitar la copia en un examen es fundamental a la hora redactar las preguntas, cualquier método basado en la memoria es poco eficiente y mejorable (Fernando G. Valderrama, 2010), y sin embargo siguen apareciendo los exámenes tipo test. Sin lugar a dudas es el método más cómodo cuando se trata de evaluar a cientos de alumnos como es el caso de los cursos masivos online, pero evitar la copia, o las posibilidades de trampa en uno de estos exámenes resulta cada vez más difícil. Tras ver algunos de estos métodos de generación de números aleatorios, parece bastante sencillo implementar un algoritmo para nuestros test y, aunque hoy en día existen multitud de herramientas que nos generan estos números *(ver Ilustración 2)* resulta muy interesante conocer el funcionamiento de estos procedimientos, no solo para la creación de las plataformas en los cursos masivos online, sino para darse cuenta del importante problema que supone el análisis de las preguntas de nuestras evaluaciones.

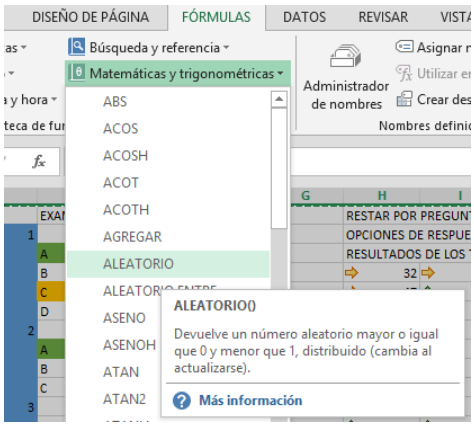


**Ilustración 1.-** Función ALEATORIO (Excel 2013)

Si se quiere implementar un sistema como el siguiente *(ver Ilustración 3),* donde existen múltiples respuestas con múltiples puntuaciones y distinto número de ellas en cada una de las preguntas, parece complejo realizarlo con algún tipo de cálculo probabilístico, sin embargo no sería complicado. Ahora bien, cuando estas preguntas se mezclan con otras donde la respuesta puede ser combinar elementos de dos listas, o desplegables con varias opciones esta tarea se vuelve mucho más complicada.

| | EXAMEN 1 | | EXAMEN 2 | | EXAMEN 3 | RESTAR POR PREGUNTA SEGÚN LA MEDIA DE RESPUESTAS POSIB | si/no |
|---|---|---|---|---|---|---|---|
| 1 | | 1 | | 1 | | OPCIONES DE RESPUESTA CON PUNTOS INTERMEDIOS: | si/no |
| | A | | A | | A | RESULTADOS DE LOS | |
| | B | | B | | B | 26 86 56 0 | |
| 2 | | | C | 2 | | 91 63 27 28 | |
| | A | | D | | A | 85 70 7 43 | |
| | B | 2 | | | B | 74 60 30 49 | |
| | C | | A | | C | 30 7 52 39 | |
| 3 | | | B | | D | 16 18 54 37 | |
| | A | | C | | E | 81 65 74 86 | |
| | B | 3 | | | F | 99 0 67 97 | |
| | C | | A | 3 | | 94 59 87 52 | |
| | D | | B | | A | 32 3 40 45 | |
| | E | 4 | | | B | 59 56 82 5 | |
| 4 | | | A | | C | 82 78 67 94 | |
| | A | | B | | D | 71 79 48 31 | |
| | B | | C | | E | 11 24 33 67 | |
| | | | | | | 60,79 47,71 51,71 48,07 | |

**Ilustración 3.-** Diseño para evaluación de test (plataforma u-mooc) (http://u-mooc.appspot.com/)

Se ha utilizado en la plataforma U-MOOC [http://u-mooc.appspot.com] para este estudio, una serie de herramientas o programas realizados en Python que en un principio servían para realizar un test de los exámenes en la plataforma para que además calculen la posibilidad de que un alumno apruebe aleatoriamente un examen, siguiendo las reglas comentadas anteriormente con un resultado exitoso. Aunque este estudio trata de facilitar este proceso a los docentes a través de programas más comunes como es el caso de Excel.

El sistema evaluaría todos los test de preguntas con una batería de pruebas aleatorias, donde puede haber distinto número de respuestas ciertas y otras que quizá no lo sean tanto, pero que tienen alguna puntuación. Se podrá restar o no las preguntas falsas, etc.

Esta batería de test nos indicará cual es la posibilidad de aprobar un examen de un modo rápido y fiable, aunque no sea con un método estadístico, pero además nos proporciona una herramienta de comprobación para la plataforma que permite solventar cualquier error de implementación. Además, con gran sencillez, posibilita incluir diferentes opciones que de un modo estadístico resultan mucho más complejas.

Del profesor depende entonces el ampliar o no el número de algunas respuestas, o la cantidad de preguntas de alguno de los test para obtener una probabilidad lo más baja posible de que un alumno consiga su objetivo por métodos aleatorios.

## CONCLUSIONES

Existen diversas herramientas para generar plataformas de cursos masivos online o MOOC, entre ellas course-builder [https://code.google.com/p/course-builder], Coursera [https://www.coursera.org], EdX [https://www.edx.org], etc. Y aunque en algunas de ellas es posible diseñar test para comprobar la fiabilidad de la plataforma, ninguna permite aún al profesor realizar una batería de test aleatorios con las preguntas que ha diseñado.

Resulta interesante conocer el funcionamiento de estos números aleatorios y que puede hacerse con ellos, para tratar de implementar algún método que ayude a profesores y alumnos a obtener una mejor herramienta de evaluación en los test que generan las nuevas plataformas para los MOOC.

## REFERENCIAS BIBLIOGRAFICAS

Bu, R. C. (1993). *Simulación: Un enfoque práctico*. Editorial Limusa SA De CV.

Enrique Sánchez Acosta. (s.f.) *Chessprogramming.* Recuperado mayo 17, 2013, a partir de http://chessprogramming.wikispaces.com/Enrique+S%C3%A1nchez+Acosta

Ciencia explicada: *Exámenes tipo test, esperanza matemática y porqué los errores DEBEN restar.* (s.f.). . Recuperado mayo 16, 2013, a partir de http://www.ciencia-explicada.com/2010/04/examenes-tipo-test-esperanza-matematica.html

*Course-builder - Course Builder - Google Project Hosting.* (s.f.). Recuperado abril 26, 2013, a partir de https://code.google.com/p/course-builder/

*Coursera.* (s.f.). . Recuperado mayo 17, 2013, a partir de https://www.coursera.org/

*edX.* (s.f.). . Recuperado mayo 17, 2013, a partir de https://www.edx.org/

Fernández, S. F. (2006). *El azar y sus problemas.* Suma: Revista sobre Enseñanza y Aprendizaje de las Matemáticas, (51), 99–105.

Fernando G. Valderrama. (2010). *Profesor el que lo lea*. Shepa.

Herrera, A. M. M. (2000). *Números aleatorios Historia, teoría y aplicaciones.* Ingeniería y Desarollo.

Lehmer, D. H. (1951). *Mathematical methods in large-scale computing units.* Ann. Comput. Lab. Harvard Univ, 26, 141–146.

Marsaglia, G. (1968). *Random numbers fall mainly in the planes*. Proceedings of the National Academy of Sciences of the United States of America, 61(1), 25.

Matsumoto, M., y Nishimura, T. (1998). *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator.* ACM Transactions on Modeling and Computer Simulation (TOMACS), 8(1), 3–30.

*Otros métodos de generación de números pseudoaleatorios.* (s.f.). Recuperado mayo 17, 2013, a partir de http://www.um.es/or/ampliacion/node18.html

*Plataforma U-MOOC.* (s.f.). . Recuperado mayo 17, 2013, a partir de http://u-mooc.appspot.com/

Saito, M., y Matsumoto, M. (2008). *SIMD-oriented fast Mersenne Twister: a 128-bit pseudorandom number generator.* Monte Carlo and Quasi-Monte Carlo Methods 2006 (pp. 607–622). Springer.

*Simulación de sistemas.* (s.f.). . Recuperado mayo 17, 2013, a partir de http://www.virtual.unal.edu.co/cursos/ingenieria/2001870/docs_curso/capitulo3/leccion3.3.htm

*Típico de los exámenes tipo test | Tontuna.com.* (s.f.). Recuperado mayo 16, 2013, a partir de http://www.tontuna.com/13042013/tipico-de-los-examenes-tipo-test/

Vallejo, P. M. (2007). *La fiabilidad de los tests y escalas*.

Wikipedia. (2013). *Lagged Fibonacci generator — Wikipedia, The Free Encyclopedia*. Recuperado a partir de http://en.wikipedia.org/w/index.php?title=Lagged_Fibonacci_generator&oldid=554620906

Zobrist, A. L. (1970). *A new hashing method with application for game playing.* ICCA journal, 13(2), 69–73.