UNLP– curso 2025

LÓGICA E INTELIGENCIA ARTIFICIAL (LeIA)

Clase 1: Introducción al curso

Prof. Dra. Claudia Pons

LIFIA - Universidad Nacional de La Plata
CIC - Comisión de investigaciones Científicas

Los docentes

TEORÍA:

Martes, 16:00 a 18:30 hs. Aula 1-4.

Profesora Dra. Claudia Pons

PRÁCTICA:

Sábado, 10:00 a 12:00 hs. Aula 2.

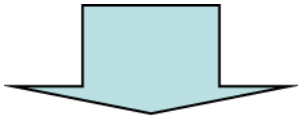
Profesora Dra. Clara Smith

Jueves, 15:00 a 17:00 hs

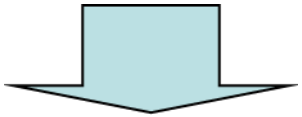
Lic. Hernán Olivera

Sobre que trata este curso? Para que me sirve?

- La importancia de LeIA para la formación del futuro profesional reside en que incentiva **una visión formal** sobre los mecanismos para construir software.
- En particular, se brindan los conocimientos y habilidades necesarios para la aplicación de métodos formales para construir software y desarrollar sistemas inteligentes.



La base formal permite **modelar rigurosamente el problema** y **razonar sobre la solución**.



De esta forma se logra incrementar la **confiabilidad y calidad del software.**

Sobre que trata este curso? Cual es su aporte específico ?

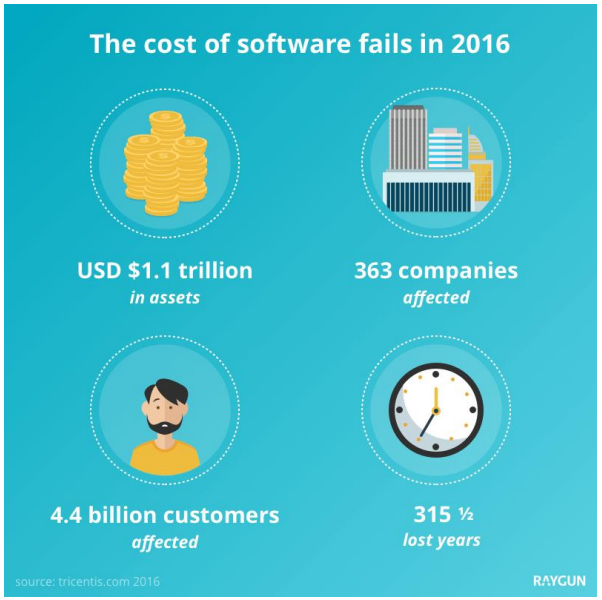
- Se brindan herramientas formales para modelar algoritmos y especificar sistemas.
- Se explica cómo usar pruebas formales y razonamientos lógicos para solucionar problemas
- Se dan las bases para la aplicación de técnicas de verificación formal a sistemas de software.
- Se analizan los beneficios potenciales de usar métodos formales de especificación y verificación de programas.
- Se brindan las bases para la comprensión de la inteligencia artificial y su aplicación práctica.
- Se analiza el paradigma de programación lógica y aprendizaje de máquina (machine learning).

Crisis del software: por que debemos preocuparnos?

- Al menos US$ 1 Trillón se pierden cada año en IT, en el mundo*
  - Proyectos abandonados
  - Rectificación de errores
  - Perdidas como consecuencia de errores y fallas

- No todo esto puede evitarse con mejores procesos
  - Errores operacionales
  - Condiciones de negocio muy cambiantes

- Ahorro potencial: al menos 2/3.



*Fuente: https://www.tricentis.com/ 2018.

Ingeniería de Software vs. otras ingenierías

- Ingeniería Civil
  - Los puentes no se caen


Reliable Engineering

Ingeniería de Software vs. otras ingenierías

- Ingeniería Civil
  - Los puentes no se caen.
- Ingeniería Mecánica
  - Los automóviles son confiables


NEXT

Ingeniería de Software vs. otras ingenierías

- Ingeniería Civil
  - Los puentes no se caen
- Ingeniería Mecánica
  - Los automóviles son confiables
- Ingeniería de Software
  - Los programas no fallan?


Rapid
A fatal exception 0E has occurred at 0028:C0011E36 in VXD VMM(01) +
00010E36. The current application will be terminated.
* Press any key to terminate the current application.
* Press CTRL+ALT+DEL again to restart your computer. You will
lose any unsaved information in all applications.
Press any key to continue _

Por que construir software es tan difícil?

- El elemento humano:
  - requerimientos inconsistentes e incompletos.
  - condiciones de negocio cambiantes.

- El elemento matemático
  - No-determinismo
    - Externo debido a los inputs
    - Interno debido a la concurrencia
  - Enorme conjunto de comportamientos
  - Aunque los requerimientos fuesen completos, no-cambiantes y formalmente especificados, es imposible analizar todos los comportamientos.

Ejemplo: algoritmo Bubble Sort

```
BubbleSort(int[] a, int n)
{
    for (i=0; i<n-1; i++)  {
        for (j=0; j<n-1-i; j++)  {
            if (a[j+1] < a[j]) {
                tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
             }
         }
    }
}
```

n	#inputs
1	2^32
2	2^64
..	
..	

Aun para programas pequeños, la enumeración de todos los comportamientos posibles es imposible!!

Un lenguaje de programación muuuuy simple

$x \in \text{Variable}$

$P \in \text{Program} = \text{assert } x \mid x\text{++} \mid x\text{--} \mid$
$P_1 ; P_2 \mid \text{if } x \text{ then } P_1 \text{ else } P_2 \mid \text{while } x \; P$

Assertion checking para este lenguaje es indecidible!

El ABC de los algoritmos de verificación de programas

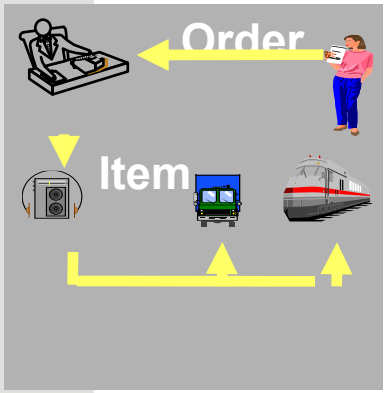
- Soundness (correctitud)
  - Si el algoritmo de verificacion reporta fallas, entonces el programa falla.
- Completeness (completitud)
- Si el programa falla, entonces el algoritmo de verificacion reporta fallas.
- Termination (terminacion)
  - El algoritmo de verificacion termina.

Es imposible cumplir este ABC, en general

¿ Cuál es el punto?

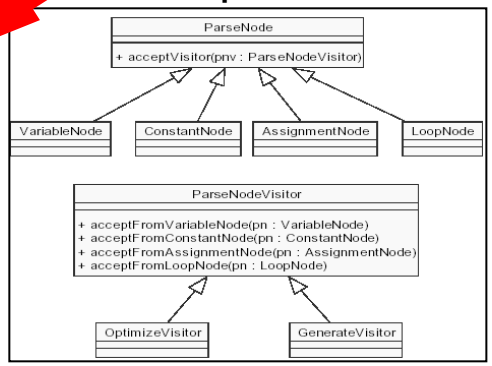
"Necesitamos resolver el problema **correcto**, correctamente"

El problema real



Users/clients

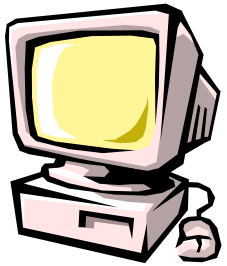
La especificación



Software engineers:
analists, designers, architects

Descripcion informal, y por lo tanto, no verificable

La implementación



Code writters

Testing,
prueba y error

¿Métodos Formales al rescate?

¿Que es un “Método Formal” en Ingeniería de Soft?

Es un “método” **bien definido** (en general matemáticamente) para construir software.

“método” = qué, quién, cuándo, cómo ?

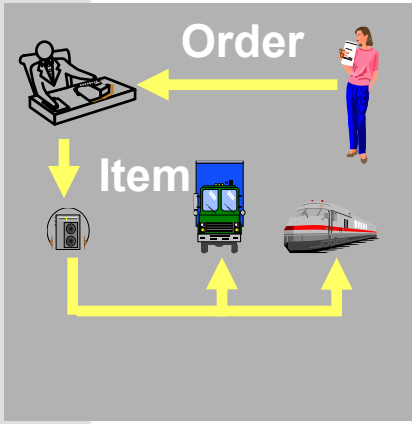
¿ Existen métodos formales en Ingeniería de Soft?

☹ **NO** exactamente…

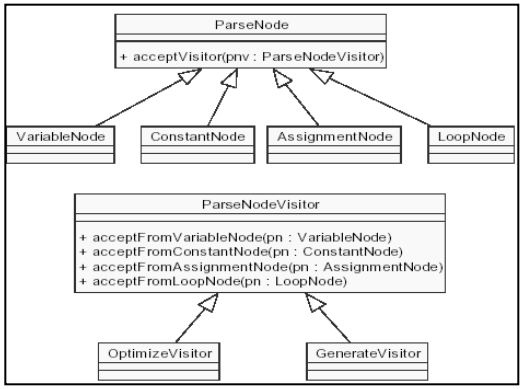
☺ **SI** existen técnicas y herramientas formales que ayudan a modelar el problema y razonar sobre la solución.

Técnicas y herramientas formales al rescate!

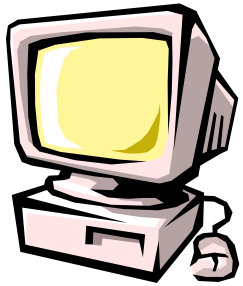
El problema real



Especificación formal



La implementación

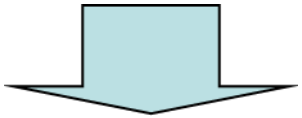


Lenguajes formales de especificación :
Z, UML/OCL, JML, Spec#
Verificación formal y derivación de propiedades

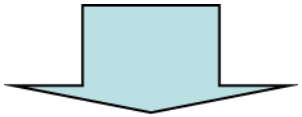
Derivación de programa (automática o semi)
Vs.
Verificación de programas

Reiteramos… Para que me sirve este curso?

- La importancia de LeIA para la formación del futuro profesional reside en que incentiva **una visión formal** sobre los mecanismos para construir software.
- En particular, se brindan los conocimientos y habilidades necesarios para la aplicación de métodos formales para construir software y desarrollar sistemas inteligentes.



La base formal permite **modelar rigurosamente el problema** y **razonar sobre la solución**.



De esta forma se logra incrementar la **confiabilidad y calidad del software.**

Bibliografía

- **BIBLIOGRAFÍA**
- Pons, Rosenfeld y Smith. Lógica para Informática. EDULP. 2017
- Hamilton, A. Logic for Mathematicians. Cambridge University Press. 1980.
- S. Russell y P. Norvig. Artificial Intelligence. A Modern Approach. Copyright © 2021 by Pearson Education. 4ta edición. 2021.

- **BIBLIOGRAFÍA COMPLEMENTARIA**
- Deep learning. I Goodfellow, Y Bengio, A Courville, Y Bengio – 2016. MIT Press.

CRONOGRAMA DE CLASES Y EVALUACIONES

Se planifican 14 clases teóricas y 14 clases prácticas.

FECHA DE INICIO: 19 /08/ 2025.

Cronograma detallado en IDEAS

Fechas de evaluación

Evaluaciones previstas	Fecha
1er parcial	02 de Noviembre de 2024
recuperatorio	16 de Noviembre de 2024
2do parcial (entrega de proyecto)	21 de Diciembre de 2024
Recuperatorio (entrega de proyecto)	3 de febrero 2024

REDICTADO

Al re-dictado de la materia en el año siguiente podrán inscribirse los alumnos que hayan cursado y desaprobado la materia en el año corriente, y que cumplan el requisito:

- Haber desaprobado alguna evaluación con nota superior a 2.