ANALISIS, DISEÑO Y CONSTRUCCION DE UNA COLECCION DE COMPONENTES DE SOFTWARE PARA ALGORITMOS GENETICOS

Grupo de Investigación en Arquitecturas de Software - ARQUISOFT

Grupo de Investigación Laboratorio De Automática e Inteligencia Computacional – LAMIC

UNIVERSIDAD DISTRITAL "FRANCISCO JOSÉ DE CALDAS"
FACULTAD - INGENIERIA DE SISTEMAS

Author:
Leidy Patricia Garzon Rodriguez

Advisers:
Eng. Henry Alberto Diosa, Ph.D.
Eng. Sergio Rojas Galeano, Ph.D.

Reviewer:
Eng. Ana María Peña Reyes, Ph.D.

[Self elaboration]

GOLDEMBERRY 2.0

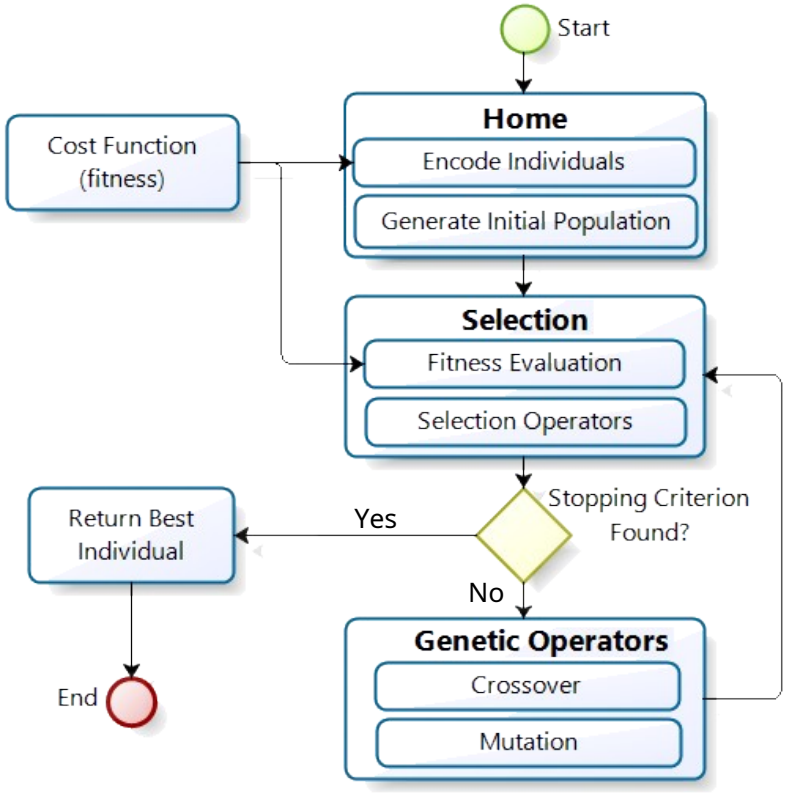
[Taken from [8]]

GOLDEMBERRY 1.0

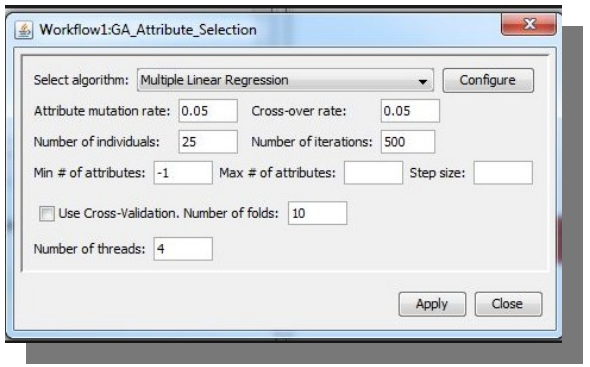
Agenda

- Introduction.
- Goals.
- Goldenberry 2.0.
- Software components.
- Methodology.
- Results.
- Conclusions.
- Future work.
- Questions.

Introduction - GA



Genetic Algorithms Flowchart [1,3]



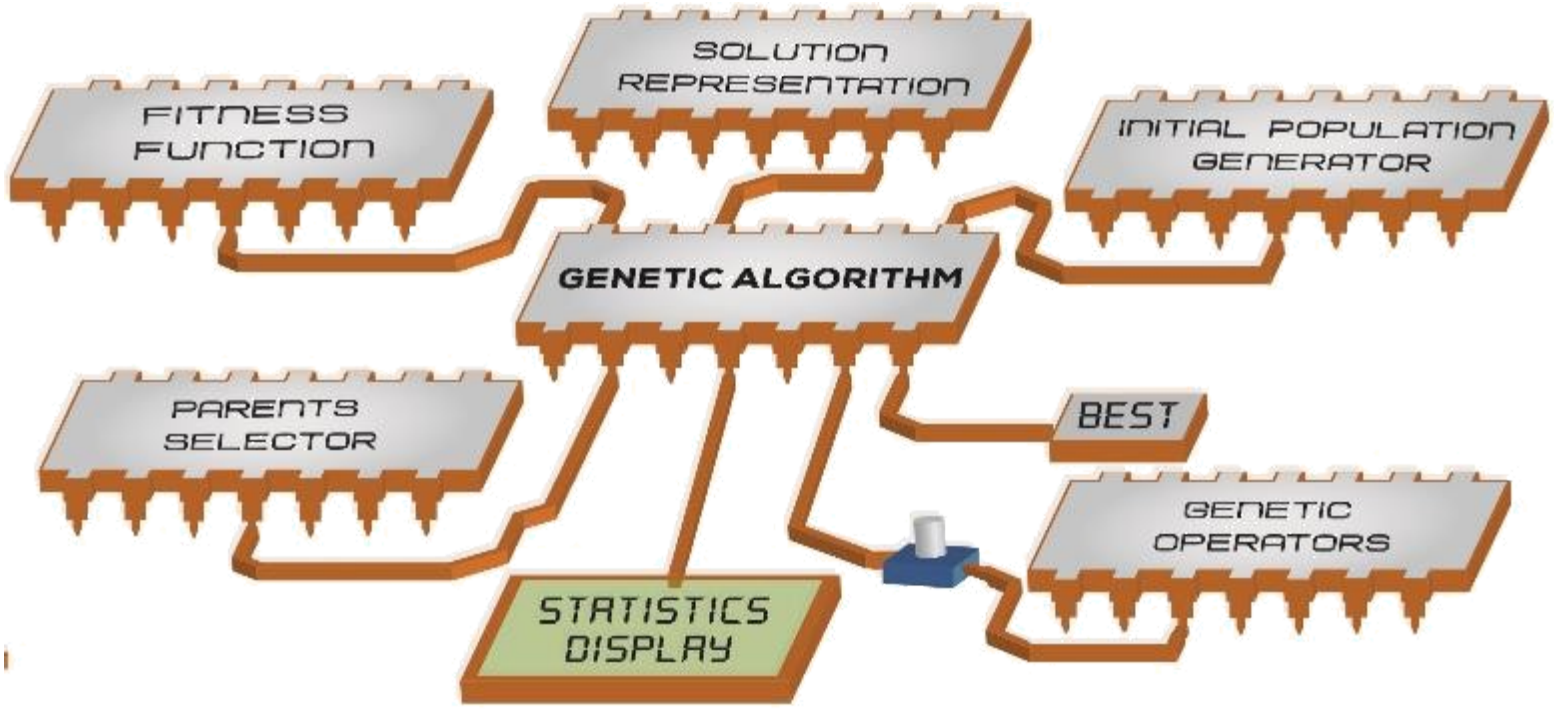

```
package examples;

import org.jgap.Chromosome;
import org.jgap.FitnessFunction;
```

```
virtual GaChromosomePtr operator ()(
    const GaChromosome* parent1,
    const GaChromosome* parent2) const;
```

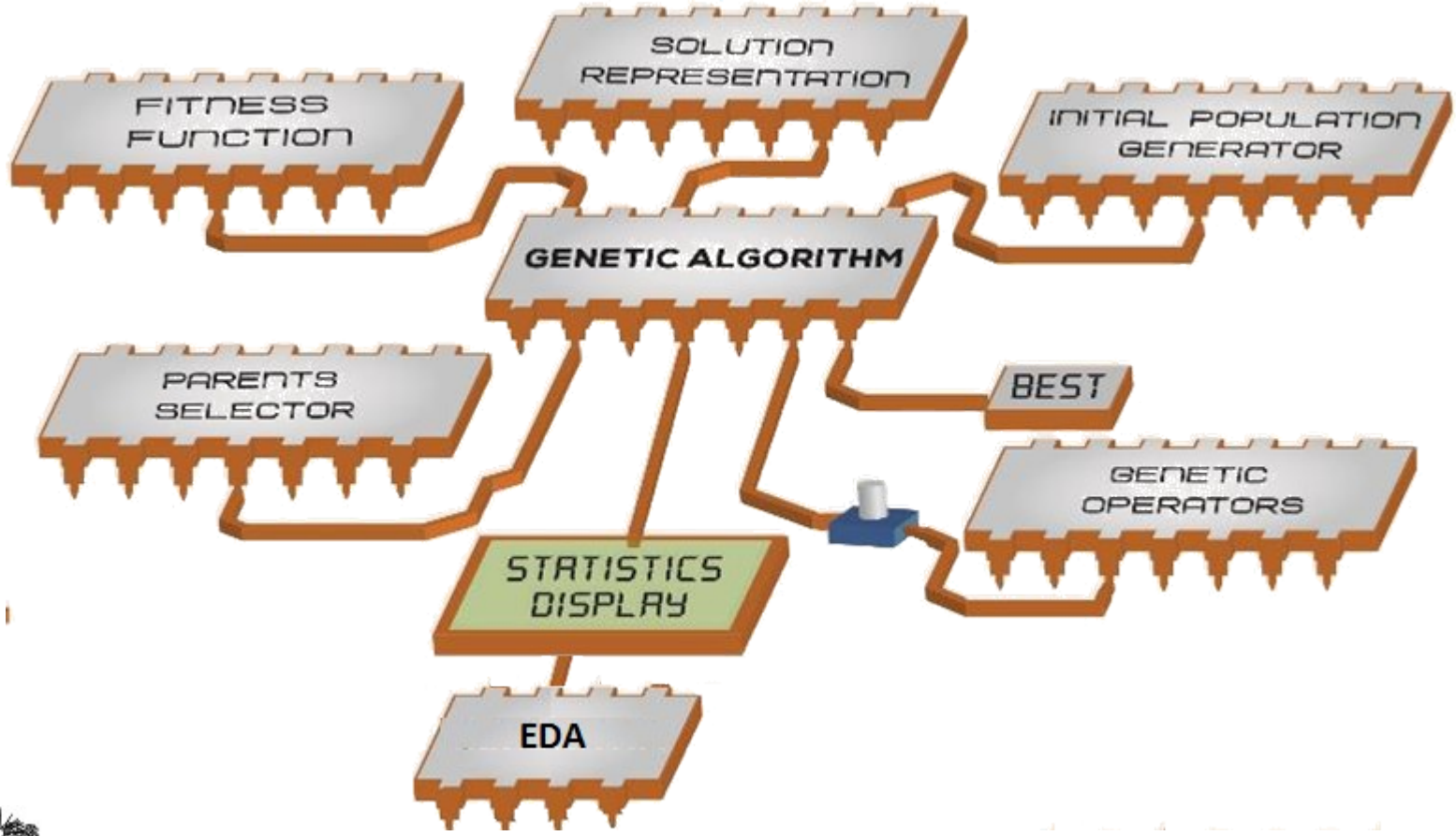
Taken from [9,11]

Introduction - GA



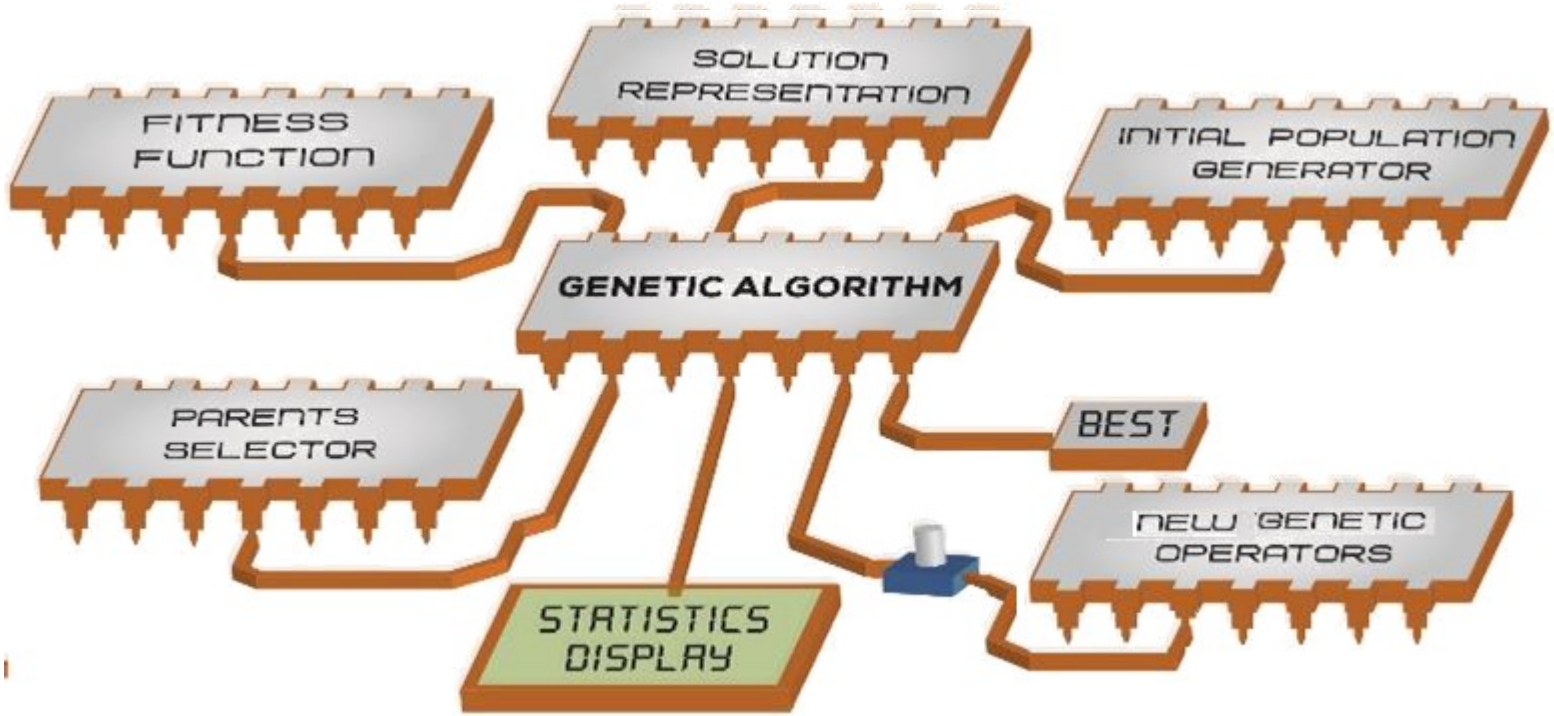
[Self elaboration]

Introduction - GA



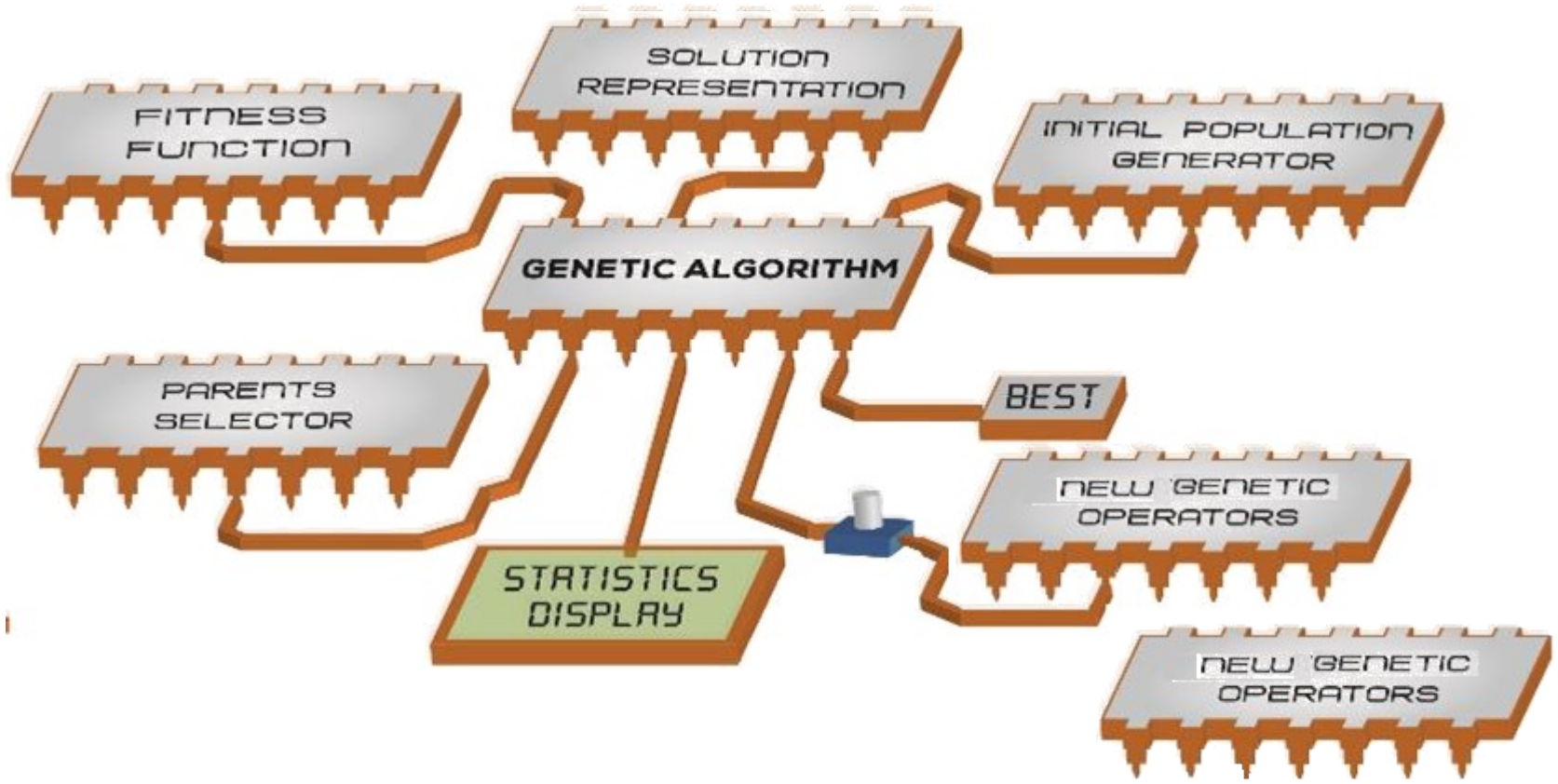
[Self elaboration]

Introduction - GA



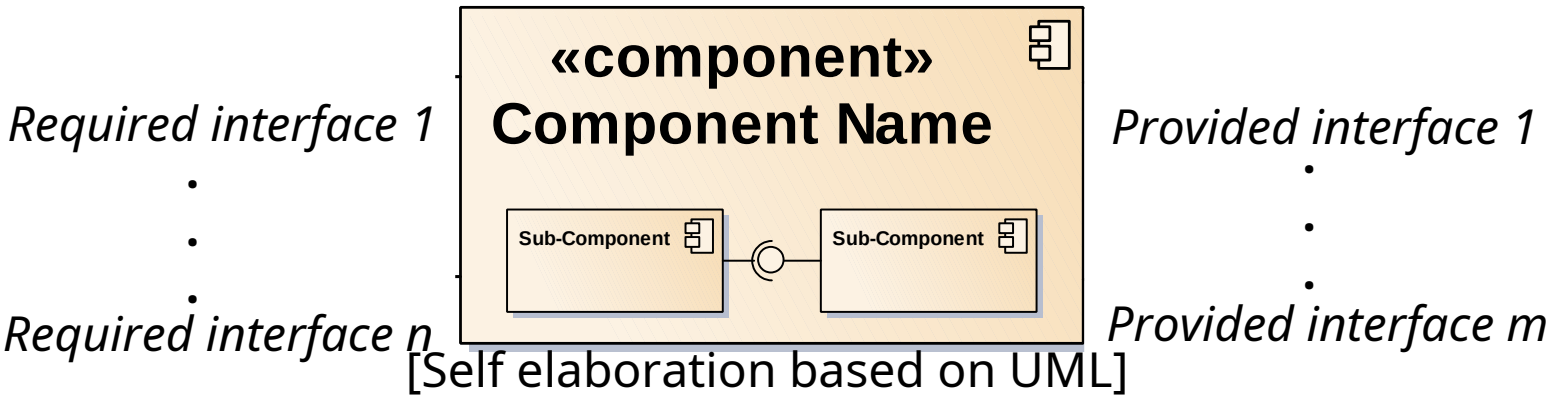
[Self elaboration]

Introduction - GA

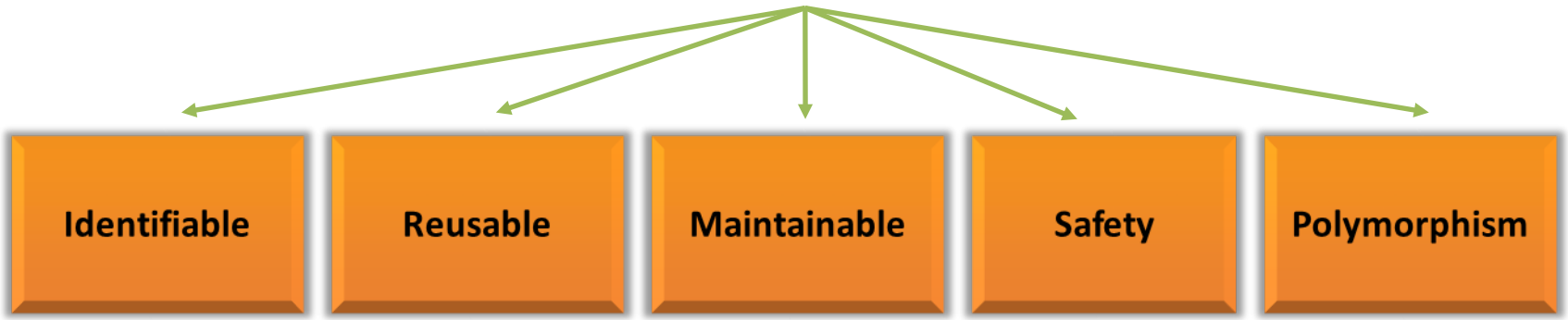


[Self elaboration]

Introduction - Software components



[Self elaboration based on UML]



Characteristics taken from [4,5]

General goal

Analysis, design and construction of a software components collection for genetic algorithms and make its implementation in an open source platform for visual programming .

Specific goals

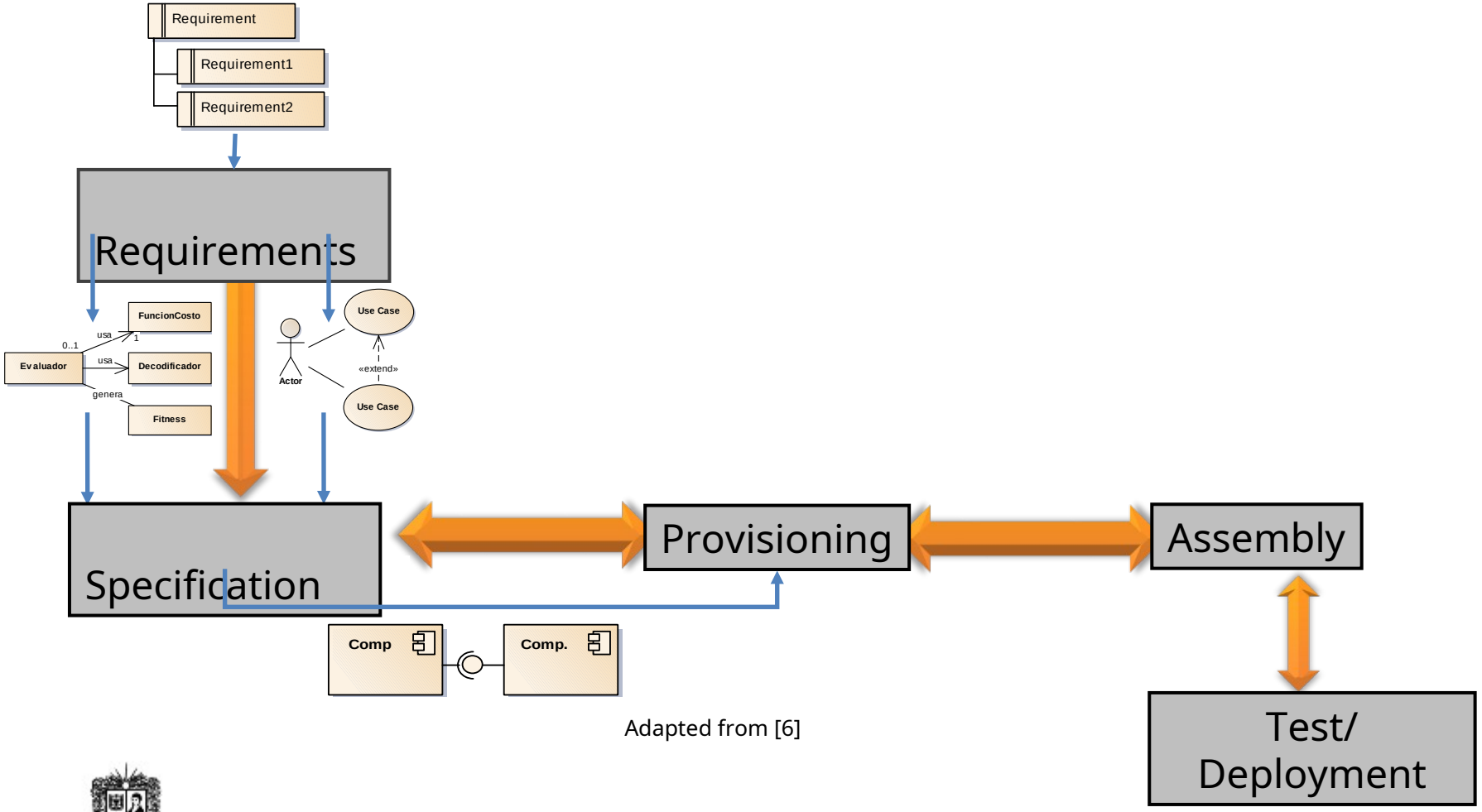
Establish	• The necessary components to build a genetic algorithm.
Analyse	• A open source framework to build the components stablished. • The functionalities that each component should offer.
Design	• The component-based software architecture, including the internal design for each component.
Implement	• The component-based software architecture proposed. • A Black box tester and calibrator component.
Distribute	• The components as add-ons from Orange, available to be downloaded from the ARQUISOFT web site (also available from the Goldenberry 1.0 web site).

Goldenberry

Acá va la demostración que consta de los siguientes pasos:

1.Ambientar sobre el canvas de orange, aclarando donde van los widgets, como están agrupados, como desplegarlos, alambrarlos, mostrando que tienen interfaces.
2.Aclarar que Orange no contaba con optimizadores y ambientar sobre GB1.
3.Mostrar los componentes de GB2 e ir uno a uno mostrando lo que se implemento, tipos de funciones de costo, mutadores, selectores, recombinadores.
4. Experimento con una función de costo de dominio binario y una continua No se si debo incluir una diapositiva con los resultados o simplemente hacer el ejercicio y nombrar los artículos usados en la experimentación.

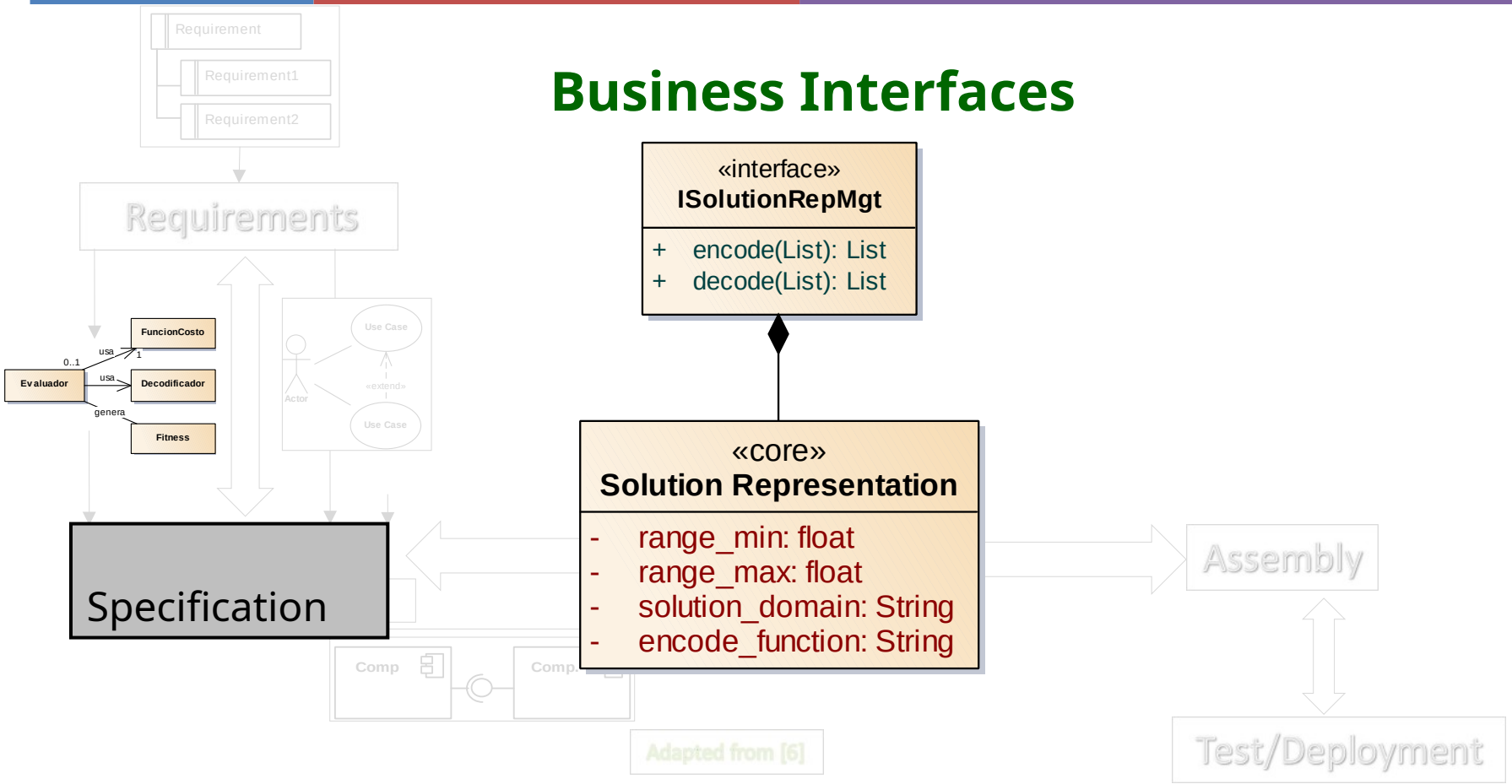
Methodology



Adapted from [6]

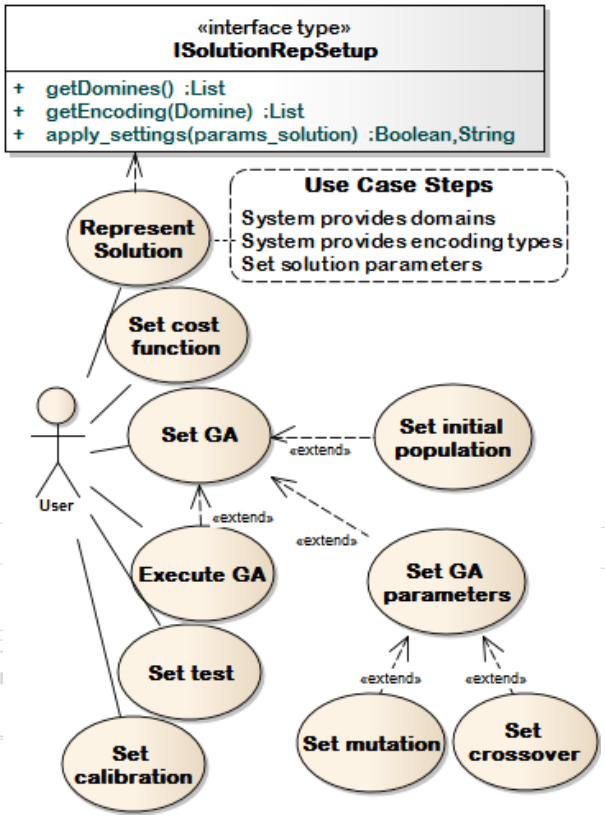
Methodology

Business Interfaces

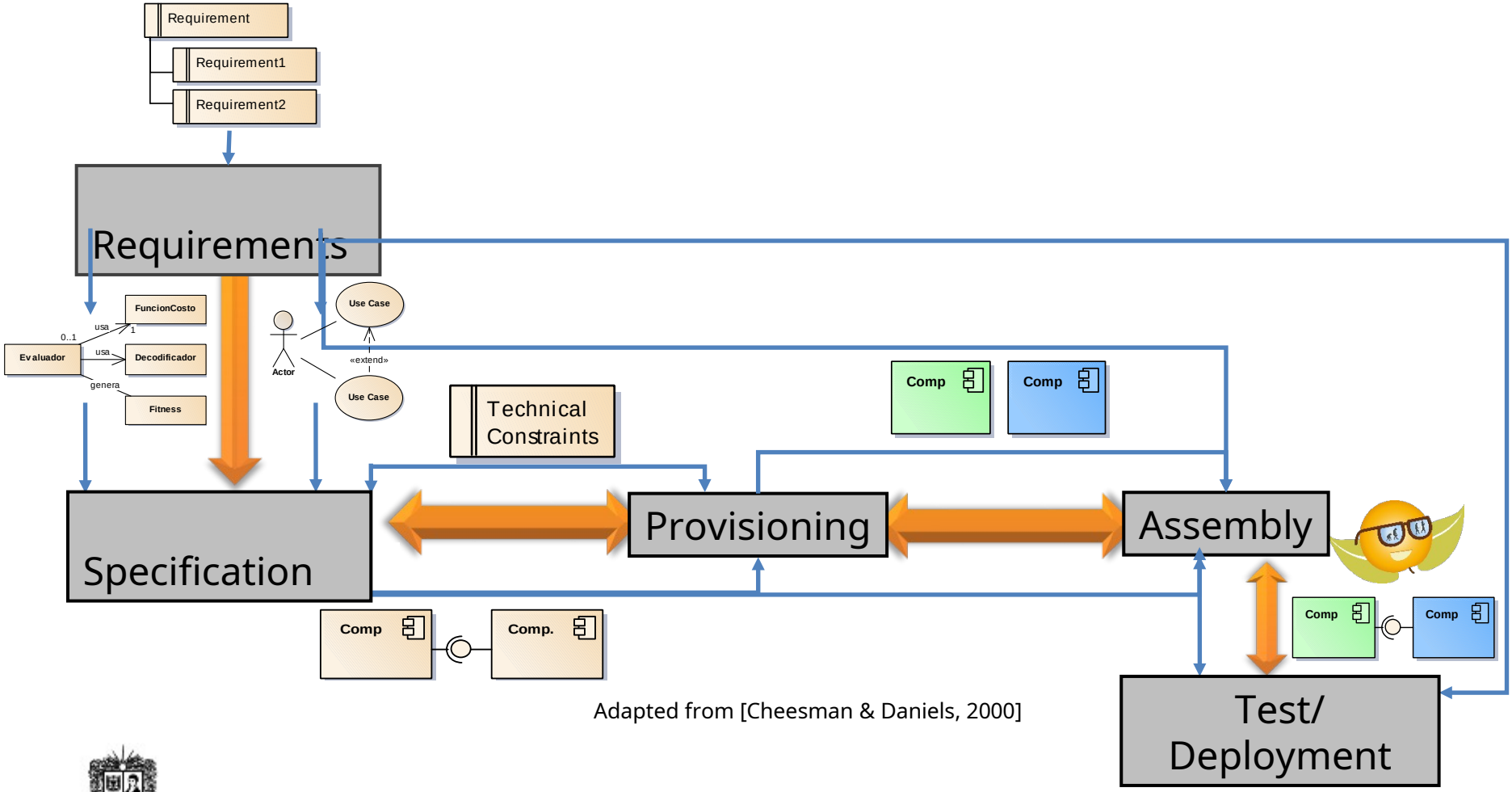


Methodology

System Interfaces



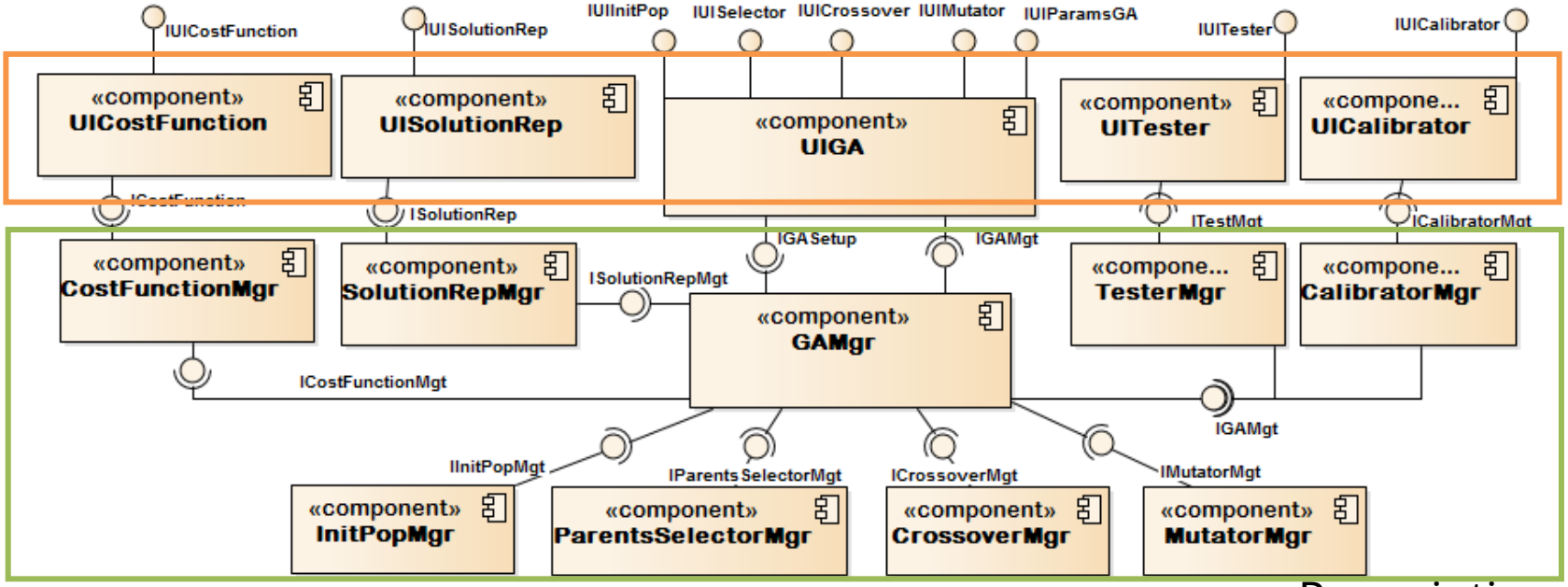
Methodology



Adapted from [Cheesman & Daniels, 2000]

Results I - Proposed Architecture

System Components

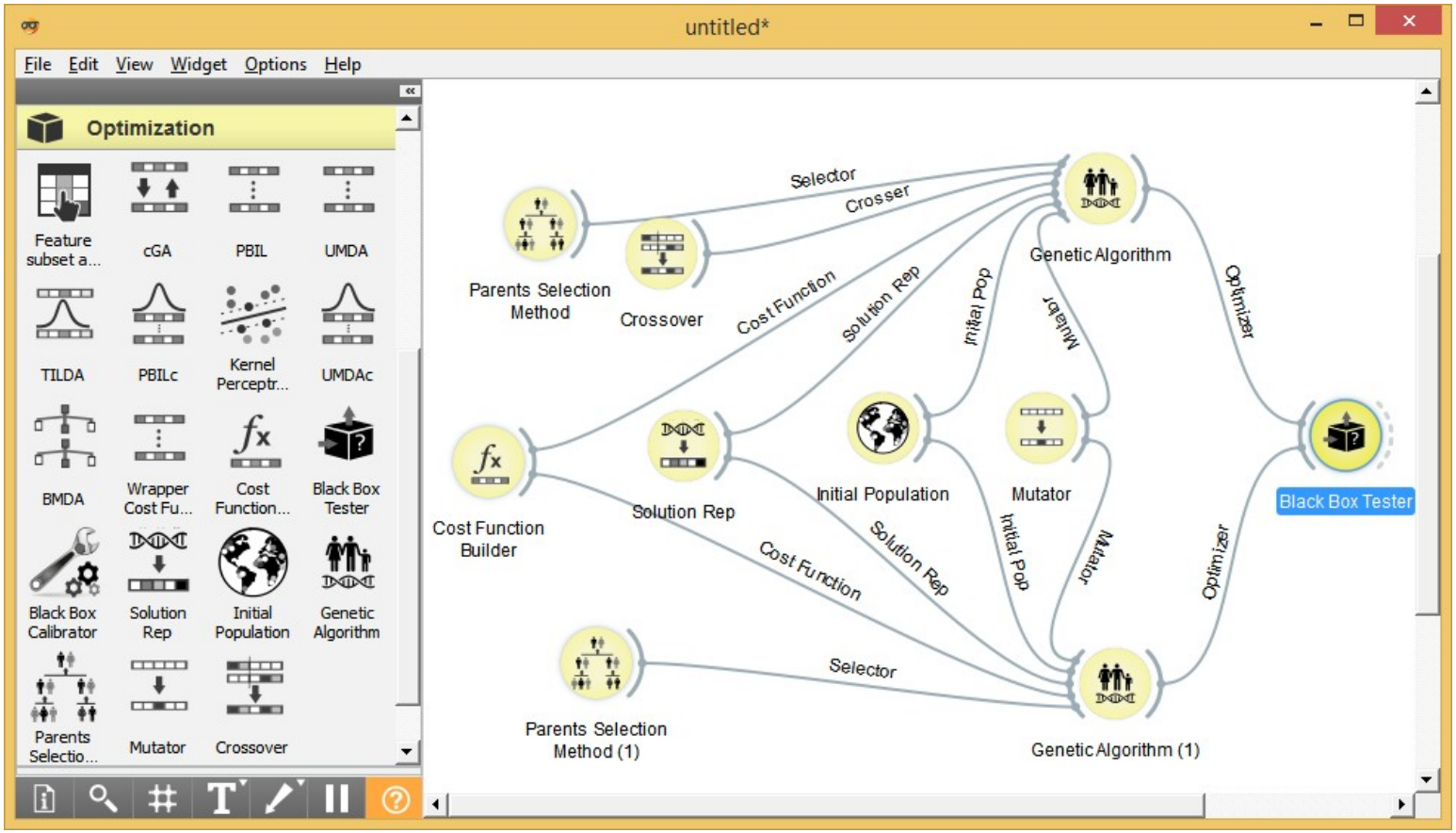


Prescriptive

Business Components

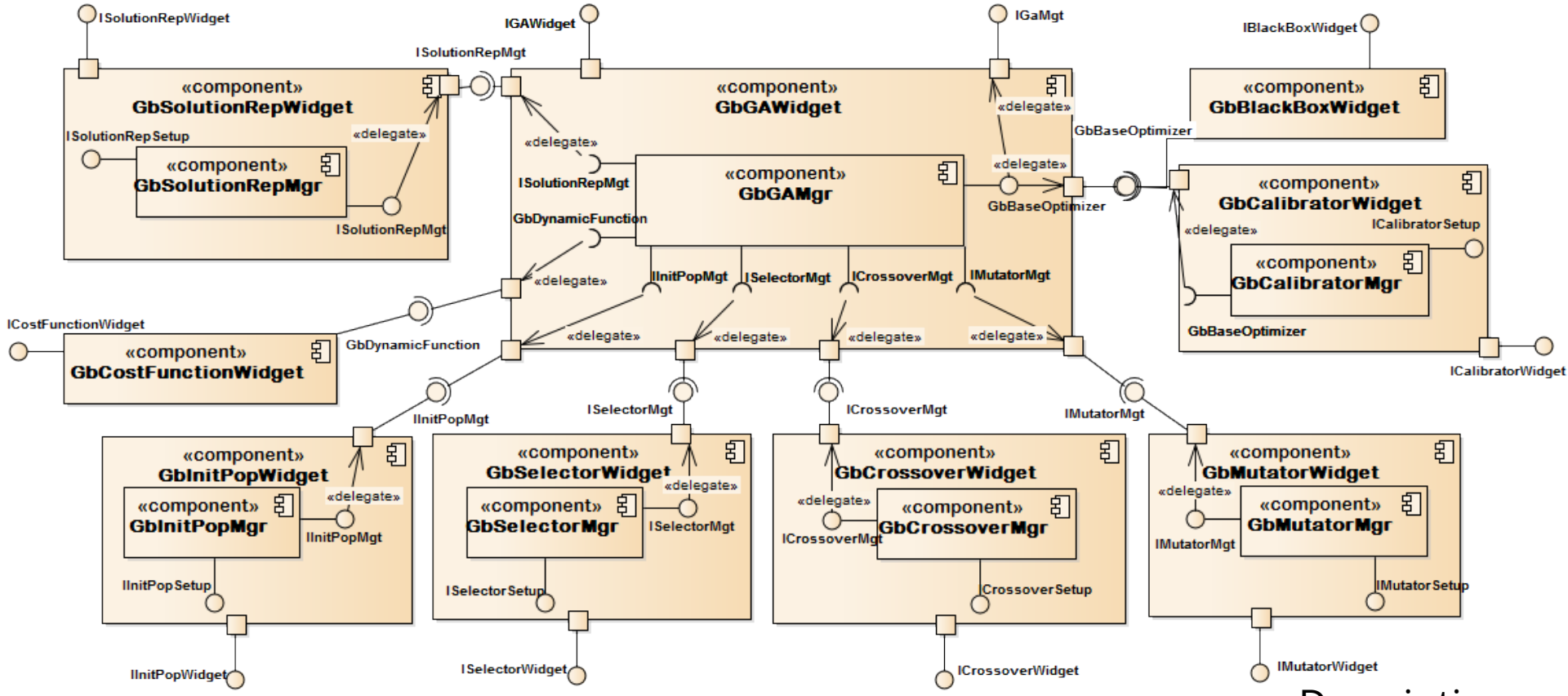
[Self elaboration based on UML]

Technical constrains



[Self elaboration]

Results II - Proposed Architecture



[Self elaboration based on UML]

Descriptive

Results – Models and Goldenberry

Requirements[58]

REQF 6.1El usuario debe contar con la opción de generar la población inicial de forma aleatoria.

REQF 6.2El usuario debe contar con la opción de generar la población inicial de forma sesgada.

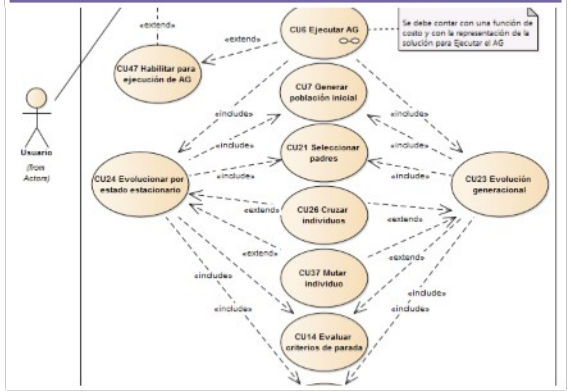
REQF 6.2.1 El usuario debe contar con un campo que le permita especificar el porcentaje de la población inicial a sesgar.

REQF 6.2.2 El usuario debe poder definir el número de sub-espacios de búsqueda e ingresar el límite máximo y mínimo de los sub-espacios; además, el valor de la media y varianza para cada sub-espacio

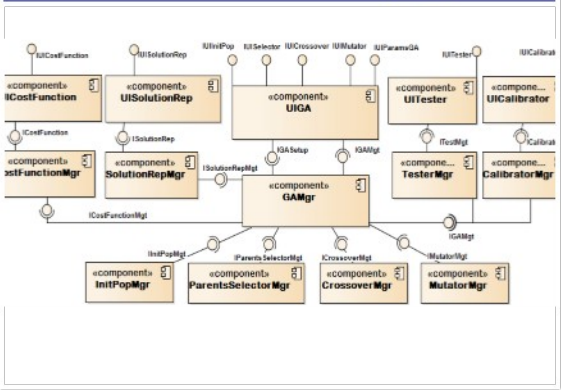
REQF 6.3El usuario debe contar con la opción de generar la población inicial introduciendo algunos individuos semilla.

REQF 6.3.1 El usuario debe contar con tres formas para el ingreso de semillas: a través de un campo que permita digitarlos, cargando un archivo plano que contenga los individuos o a través de la conexión de un componente que provea individuos

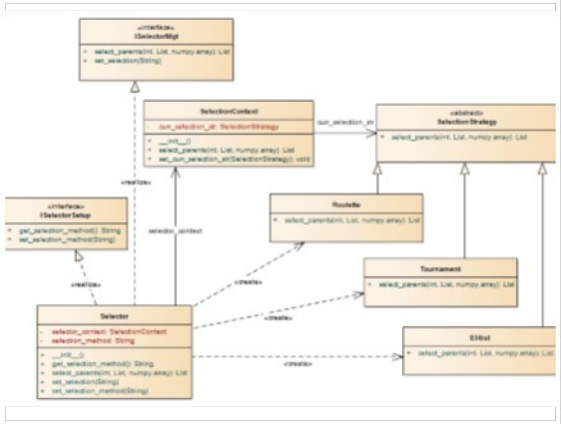
Use cases [55]



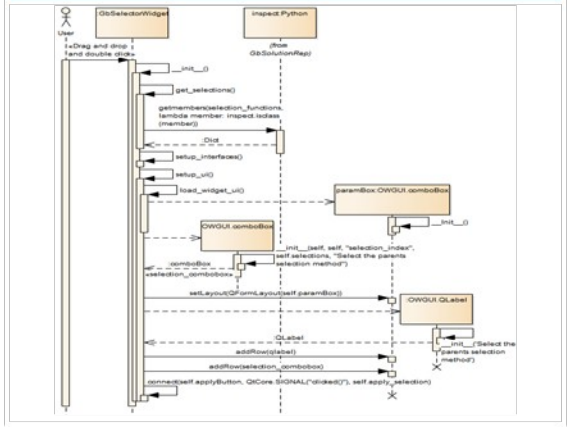
Architecture[2]



Estructural[8]



Dinamic



Goldenberry 2.0

Results

9CCC
Congreso Colombiano de Computación

September 3-5, 2014. Pereira, Colombia.

Towards a Component-based Software Architecture for Genetic Algorithms

Leidy Garzón Rodríguez
lpgarzonr@correo.udistrital.edu.co

Henry Alberto Diosa
hdiosa@udistrital.edu.co

Sergio Rojas-Galeano
srojas@udistrital.edu.co

Universidad Distrital Francisco José de Caldas
Bogotá, Colombia

***Abstract*—We are motivated on the idea of whether a component-based software architecture for evolutionary algorithms would be feasible and advantageous. We believe that depending on the evolutionary computation model, software implementing these algorithms can be robustly built assembling loosely-coupled computational blocks, likewise hardware systems that are built gluing together prefabricated electronic components. We set about to develop an initial architecture with a focus on the genetic algorithm. The paper describes the analysis and design principles used, the obtained architecture, the resulting component specification and closes with a discussion about the benefits of this approach, as well as initial steps towards its implementation in a user-friendly platform for component-based visual programming. The complete portfolio of software models is available at:**

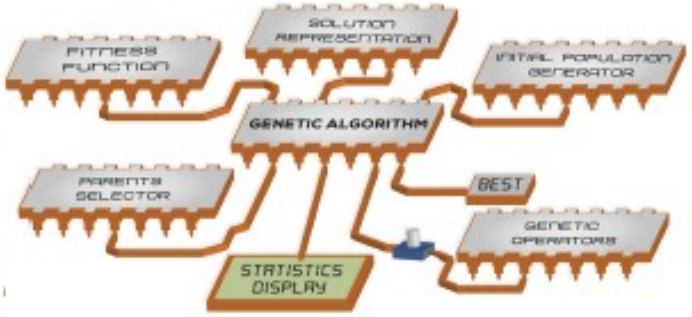


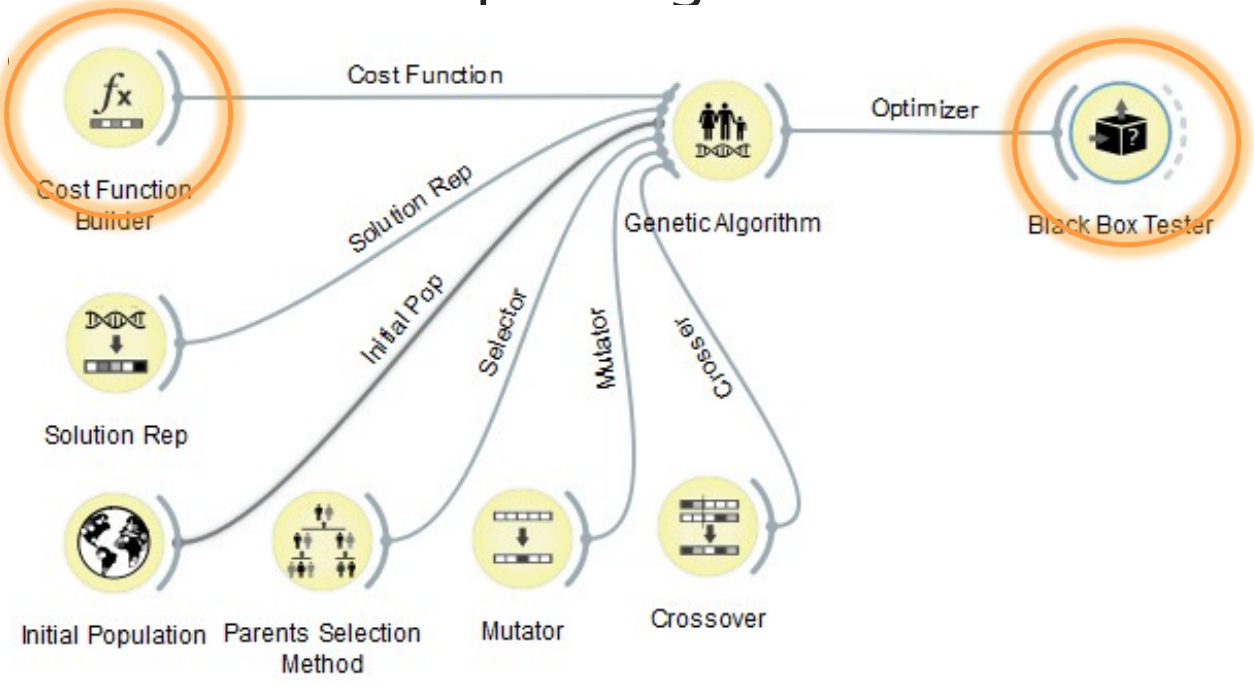
Figure 1. A fictitious depiction of an electronic component-based GA

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Facultad de Ingeniería, proyecto curricular de ingeniería de sistemas

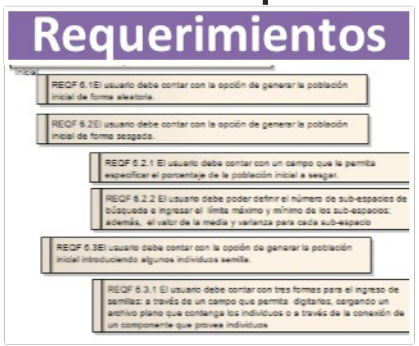
Conclusions

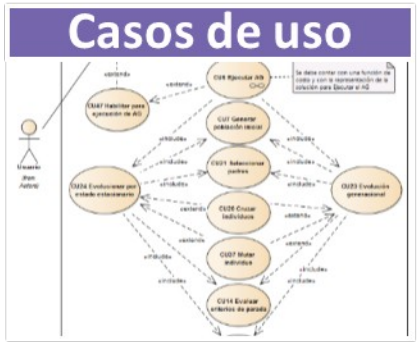
- Component-based software development is a suitable paradigm for extensible

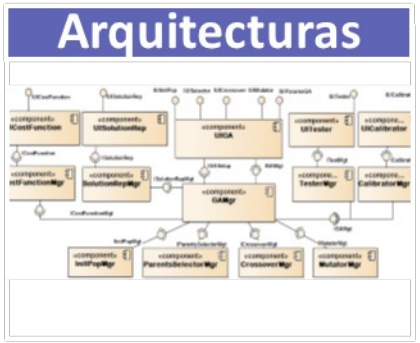


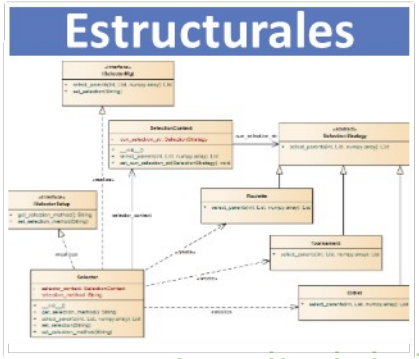
Conclusions

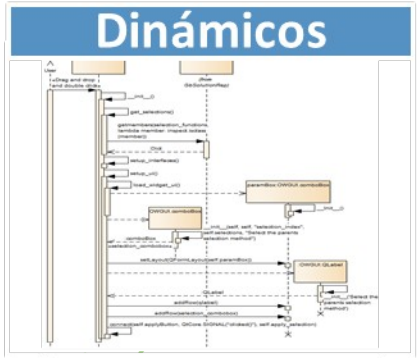
- We have described a feasible architecture for general purpose component-based GAs. The architecture is open!! and independent of runtime or implementation decisions .

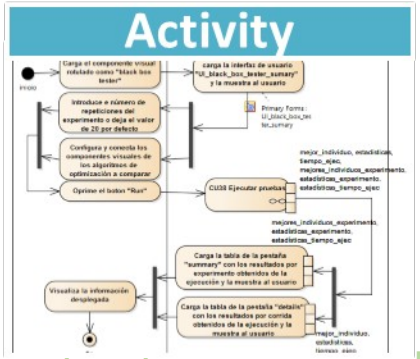












Conclusions

- The modularity and detail exhibited by this architecture makes it easily extensible to other kinds of stochastic search population-based optimization algorithms through the reuse of GB Components. And to other genetic operators due to the use of the strategy software design pattern.

Future Work

Extend the parent selector, crossover and mutation components

- Add new selection methods as range and truncate.
- Add uniform, MMAX and multipoint crossover.
- Add polynomial mutation.

Include the discrete solution domains

- Crossover PMX, CX, OX1, POS.
- Mutation EM, DM, ISM, IVM

New evolutive components

- Simulated annealing
- Hill climbing
- Diferential evolution
- Grid GA

Concurrence

References

- [1] Holland, J. (1975). Adaptation in Natural and Artifcial Systems. Ann Arbor, MI, USA: University of Michigan Press.
- [2] Holland, J. (1992). Adaptation in natural and artificial systems. Cambridge, Massachusetts: MIT Press.
- [3] Goldberg, D. (1989). Genetic algorithms in search, optimization and machine learning. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 1 edition.
- [4] Szyperski, C. (1996). Independently Extensible Systems Software Engineering Potential and Challenges. Brisbane, Australia: School of Computing Science - Queensland University of Technology.
- [5] Szyperski, C. (1998). Component Software. Beyond Object-Oriented Programming. Brisbane, Australia: Addison-Wesley, 2 edition.

References

- [6] Cheesman & Daniels, 2000] Cheesman, J. & Daniels, J. (2000). UML Components: A Simple Process for Specifying Component Based Software. Component Software Series. Londres: Addison Wesley.
- [7] Demsar, J. & Zupan, B. (2004). Orange: From experimental machine learning to interactive data mining. In Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (pp. 537-539). New York, USA: Springer-Verlag New York, Inc.
- [8] Rojas-Galeano, S. & Rodriguez, N. (2013). Goldenberry: Eda visual programming in orange. In Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion, GECCO 13 Companion (pp. 1325-1332). Amsterdam, The Netherlands: ACM.
- [9] K. Meffert and N. Rotstan. Jgap, java genetic algorithms package. Available at http://jgap.sourceforge.net.
- [10] B. Dmitry. Open beagle: a generic framework for evolutionary computations. Genetic Programming and Evolvable Machines, 12(3):329–331,September 2011.
- [11] M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich. Opt4J - A Modular Framework for Meta-heuristic Optimization. In Proceedings of GECCO 2011, pages 1723–1730, Dublin, Ireland, 2011.

Questions?


DRAFT 1
REVISE.
DRAFT 2
REVISE.
JORGE CHAM © 2007
...
DRAFT 00
REVISE.
DRAFT APPROVED!
I'M SICK OF READING THIS. JUST TURN IT IN.
...
WWW.PHDCOMICS.COM

Thank you!