Interactive multi-objective evolutionary optimization of software architectures

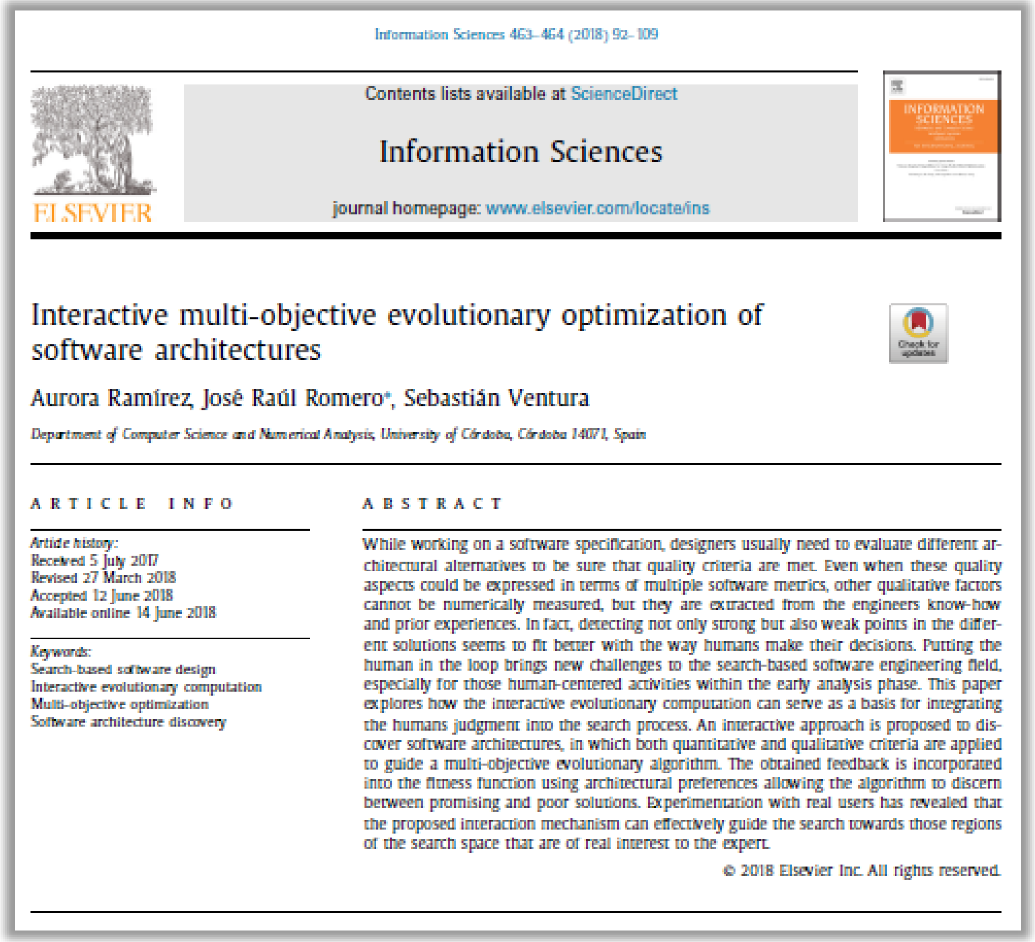
Aurora Ramírez, José Raúl Romero, Sebastián Ventura

Dept. Computer Science and Numerical Analysis. University of Córdoba.

Track Ingeniería del software guiada por búsqueda
Jornadas de Ingeniería de Software y Bases de Datos
Cáceres, 4 de septiembre de 2019

Content

A. Ramírez, J.R. Romero, S. Ventura. "Interactive multi-objective evolutionary optimization of software architectures". *Information Sciences*, vol. 463-464, pp. 92-109. 2018.



Information Sciences 463–464 (2018) 92–109

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Interactive multi-objective evolutionary optimization of software architectures

Aurora Ramírez, José Raúl Romero*, Sebastián Ventura

Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba 14071, Spain

ARTICLE INFO

Article history:
Received 5 July 2017
Revised 27 March 2018
Accepted 12 June 2018
Available online 14 June 2018

Keywords:
Search-based software design
Interactive evolutionary computation
Multi-objective optimization
Software architecture discovery

ABSTRACT

While working on a software specification, designers usually need to evaluate different architectural alternatives to be sure that quality criteria are met. Even when these quality aspects could be expressed in terms of multiple software metrics, other qualitative factors cannot be numerically measured, but they are extracted from the engineers know-how and prior experiences. In fact, detecting not only strong but also weak points in the different solutions seems to fit better with the way humans make their decisions. Putting the human in the loop brings new challenges to the search-based software engineering field, especially for those human-centered activities within the early analysis phase. This paper explores how the interactive evolutionary computation can serve as a basis for integrating the humans judgment into the search process. An interactive approach is proposed to discover software architectures, in which both quantitative and qualitative criteria are applied to guide a multi-objective evolutionary algorithm. The obtained feedback is incorporated into the fitness function using architectural preferences allowing the algorithm to discern between promising and poor solutions. Experimentation with real users has revealed that the proposed interaction mechanism can effectively guide the search towards those regions of the search space that are of real interest to the expert.

Introduction

- Software design is a human-centered task
- Qualitative aspects are difficult to quantify
- The engineer should be involved in the the optimization process
- Sometimes it is easier to identify a bad solution rather than a good one

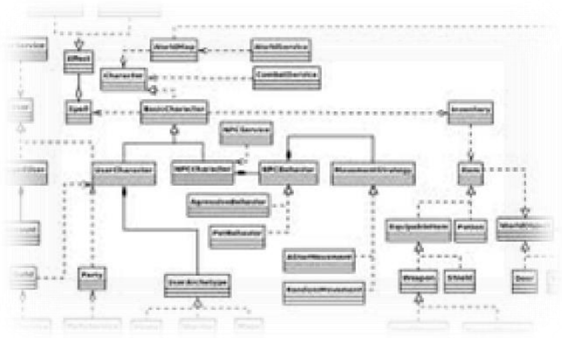
Introduction

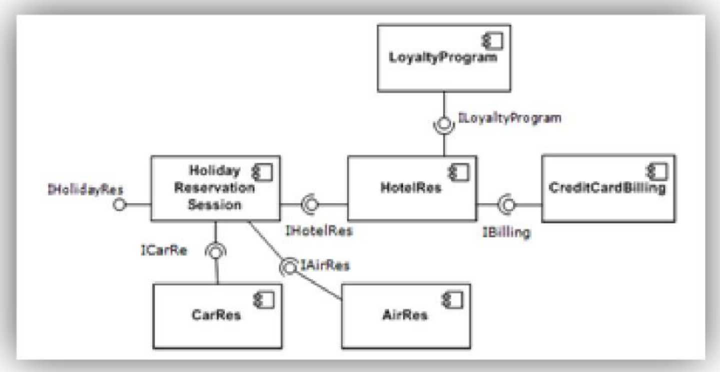
RQ1: How can the qualitative judgement of the engineer be integrated into the evolutionary discovery of software architectures?

RQ2: Does putting the human in the loop involve a significant improvement compared with not considering him/her along the optimization process?

Background
Architecture discovery

To identify the underlying **component-based architecture** of the system from the analysis information contained in a UML class diagram





<u>**Component:**</u> a cohesive group of classes that work together to satisfy the expected behavior of the component

<u>**Interface:**</u> a directed relationship between classes belonging to different components

<u>**Connector:**</u> The linkage between required/provided interfaces

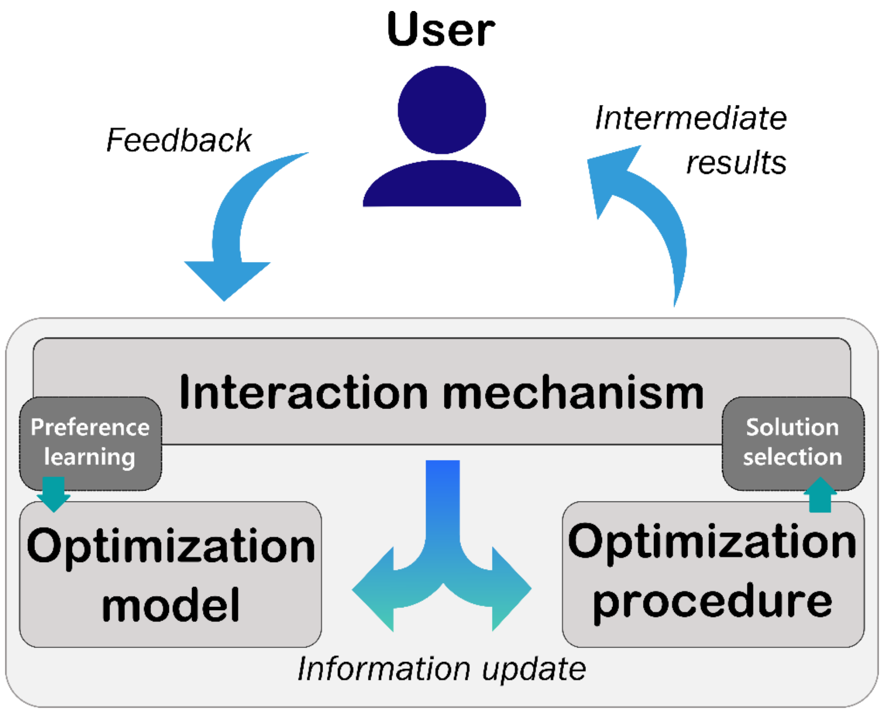
Background
Interactive optimization

Full automation is not always realistic:
- Uncertainty scenarios
- Creative tasks

Human is "put in the loop":
- To provide problem knowledge
- To increase trust on automatic results
- To meet user's expectations

Design and implementation decisions:
- Selection of solutions
- Type of feedback
- Frequency of interaction
- Information lifetime

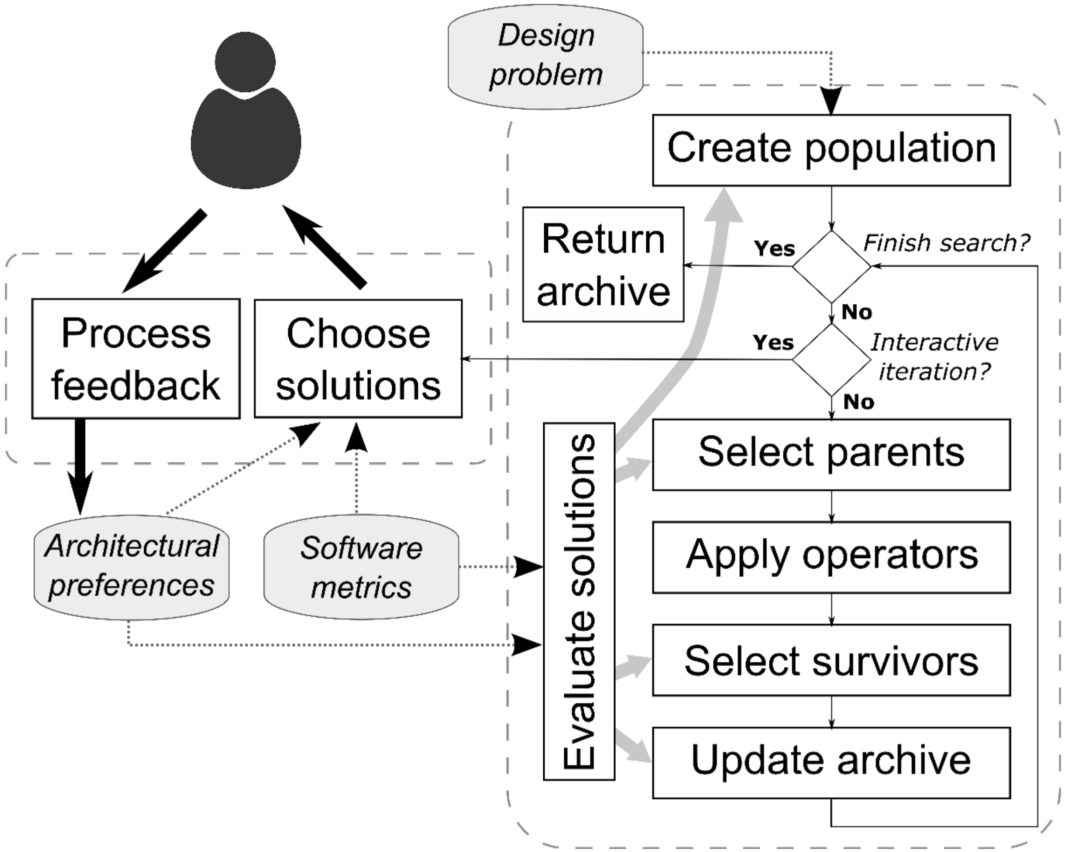
Any optimization method, including metaheuristics, in which the human **actively participates** in any step of the process to provide feedback



Interactive multi-objective evolutionary optimization of software architectures. ISGB@JISBD'19.

Interactive algorithm
Overview

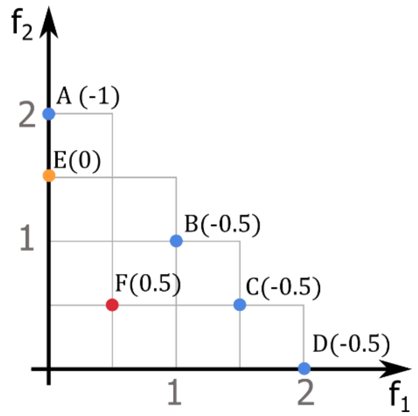
- Steady-state algorithm: two offspring are produced in each generation
- Archive to store a small set of representative solutions
- Binary tournament selection from population and archive
- Offspring replace population members with worst fitness



Interactive algorithm
Fitness function

A novel **fitness function** that combines objective and subjective evaluation criteria

$$fitness(s) = w_{obj} \cdot f_{obj}(s) + w_{sub} \cdot f_{sub}(s)$$

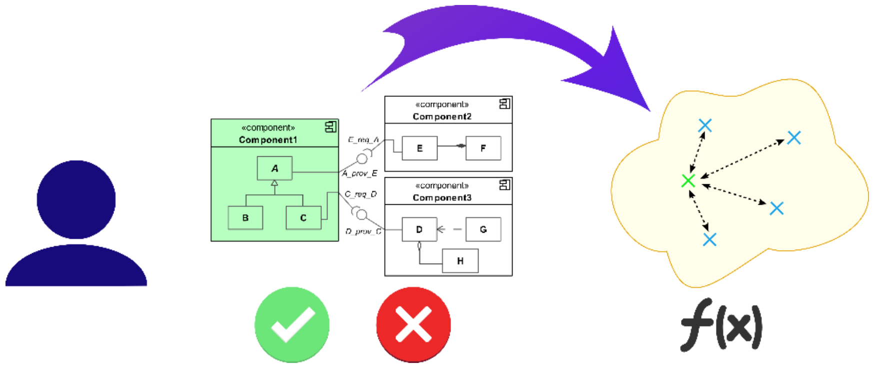


Quantitative: *Maximin* function to quantify both dominance and diversity (software measures)

$$f_{obj}(s) = \frac{1 + max_{z \neq s}(min_k(f_k^s - f_k^z))}{2} \quad \forall z \in Z$$

Qualitative: Engineer's preferences on phenotypic aspects of the solution (architectural preferences)

$$f_{sub}(s) = 1 - \frac{1}{P} \cdot \sum_{p=1}^{P} w_p \cdot pref_p(s)$$

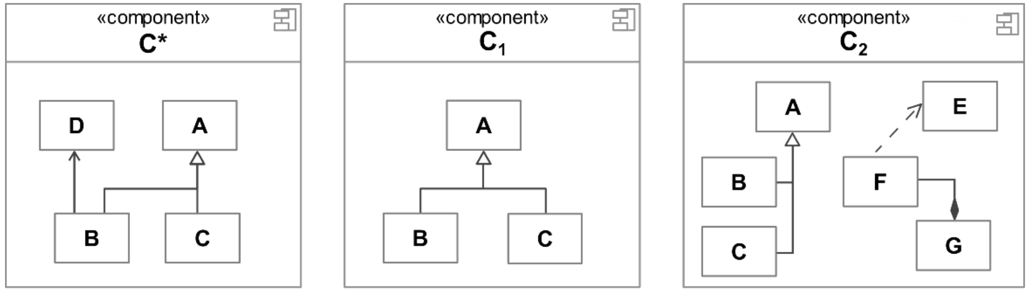


Interactive algorithm
Architectural preferences

Preference	Description
Best component	Similarity to the set of classes
Worst component	Dissimilarity to the set of classes
Best provided interface	Similarity to the set of operations
Worst provided interface	Dissimiliraty to the set of operations
Number of components	Distance to the preferred number
Metric in range	Distance to the midrange
Aspiration levels	Weighted distance to the reference point

$$pref_{bc}(s) = max\{J(classes(c), classes(c^*))\} \quad \forall c \in [1, n]$$

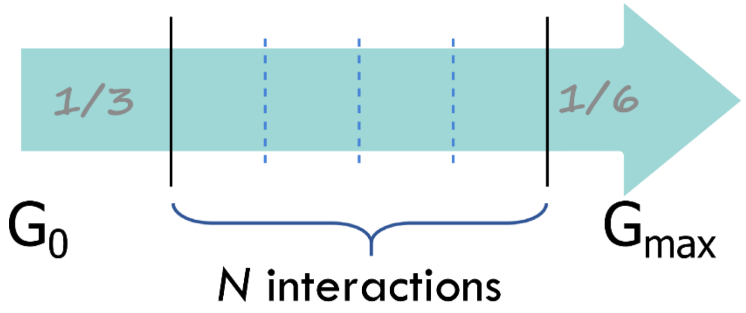
$$J(A, B) = \frac{A \cap B}{A \cup B}$$

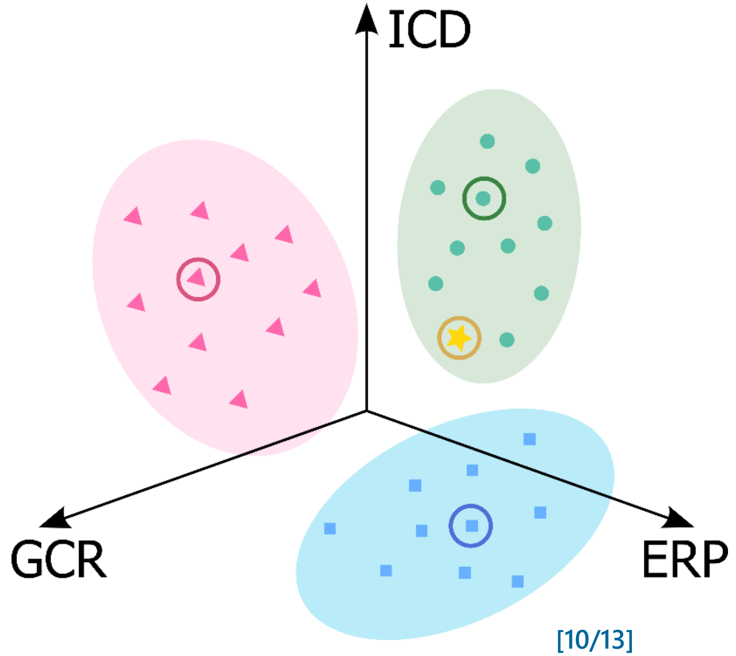


$J(C^*, C_1) = 3/4 = 0.75$ $J(C^*, C_2) = 3/7 = 0.429$

Interactive algorithm
Interaction mechanism

- Interactions are scheduled at regular intervals
- Solutions are selected from the current population using a clustering method
- The engineer rewards or penalizes some aspect of the solution by choosing one preference
- Additional actions:
  - Freeze one component
  - Add to the archive
  - Remove from the population
  - Stop the search

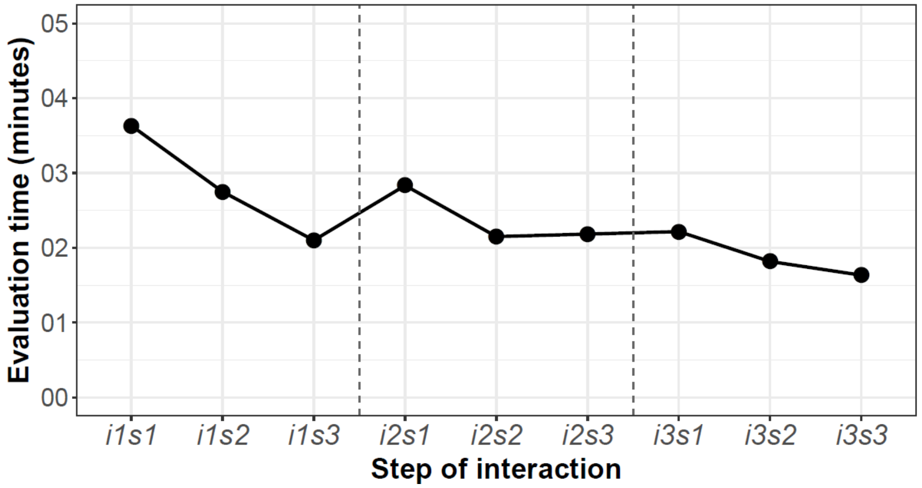




Empirical study

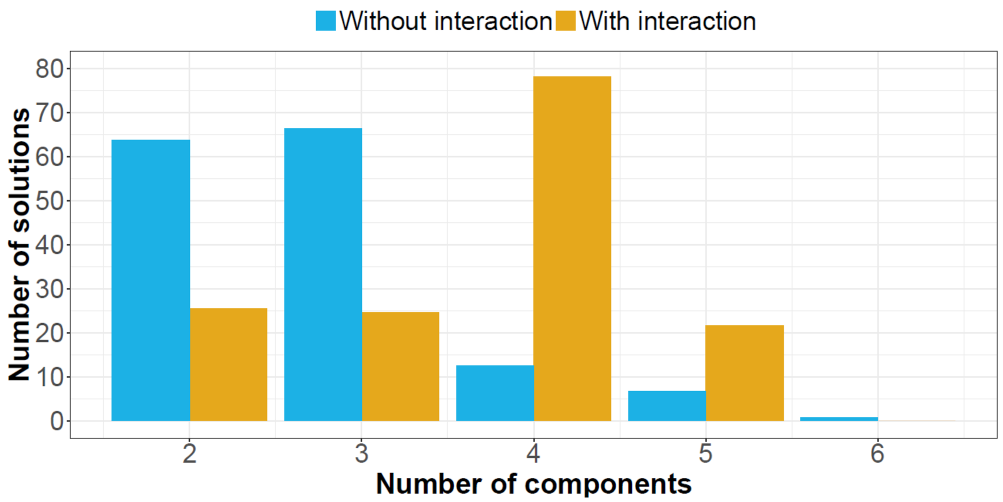
Methodology:

- Case study, 9 participants
- 3 interactions, 3 solutions each
- Log files and questionnaires



Impact of interaction:

- Cohesion is indirectly improved
- Different number of components
- Components are similar or equal to the manual design
- Actions (add, freeze) contribute to find better solutions

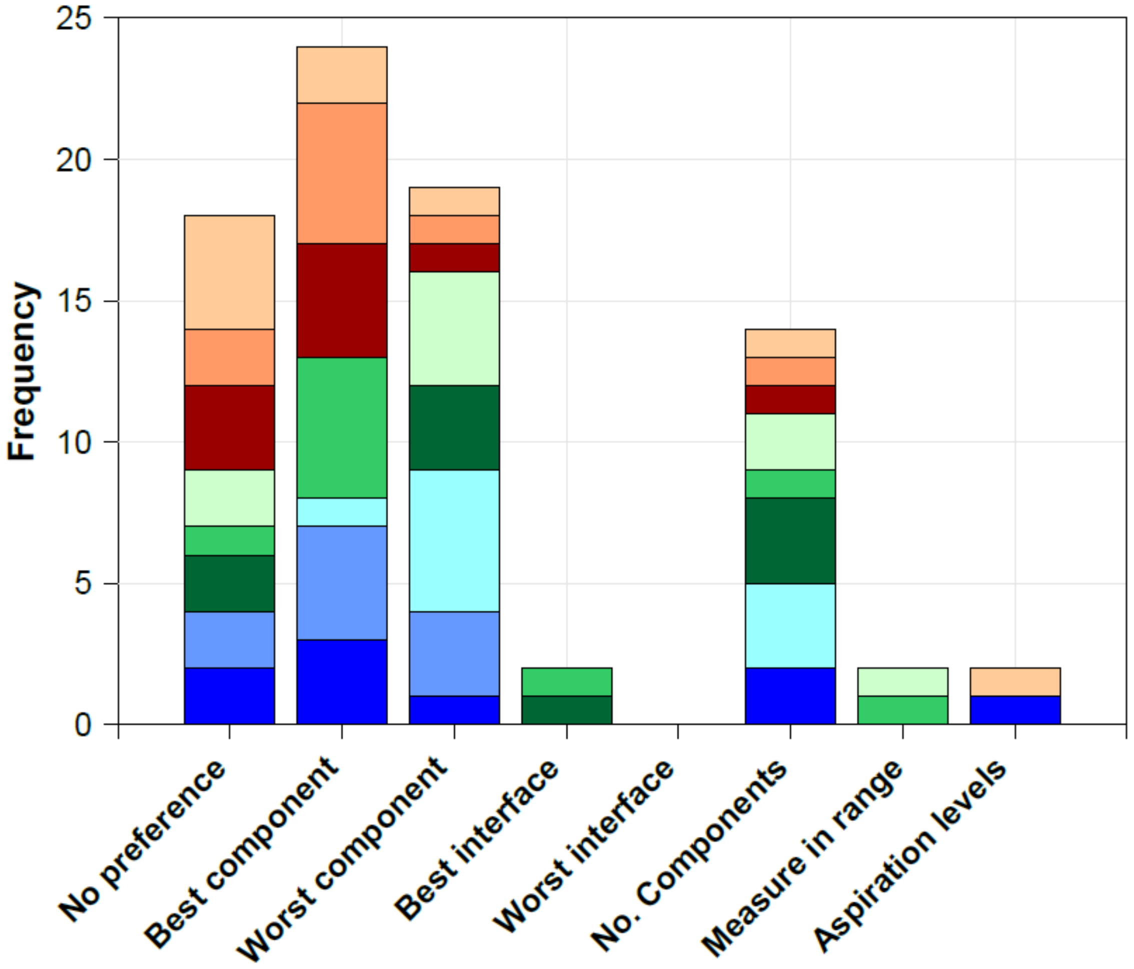


Empirical study

Analysis of architectural preferences:

- More interest in the internal structure
- Negative preferences are move frequently applied at the beginning
- Use of "no preference"

Architectural preference	Selected (%)	Useful [1-8]	Intuitive [1-8]
No preference	22.22	6.44	7.33
Best component	29,63	7.44	7.44
Worst component	23.46	7.22	7.33
Best prov. Interface	2.47	5.29	6.38
Worst prov. Interface	0.00	4.71	6.38
No. Components	17.28	7.50	7.33
Metric in range	2.47	4.17	5.44
Aspiration levels	2.47	5.80	5.22



Conclusions

- ✓ Architectural preferences as a novel mechanism for subjective evaluation
- ✓ Design measures are still needed at the beginning
- ✓ Negative options are useful too
- ✓ Humans are willing to participate (even more) in the optimization

Future work:

- ➢ Improve the interactive experience (more preferences and flexibility)
- ➢ Analyze how other human design abilities could be integrated

Interactive multi-objective evolutionary optimization of software architectures

Aurora Ramírez, José Raúl Romero, Sebastián Ventura

UNIVERSIDAD DE CÓRDOBA

Aurora Ramírez

aramirez@uco.es
https://www.uco.es/users/aramirez
@aurora_rq

Track Ingeniería del software guiada por búsqueda
Jornadas de Ingeniería de Software y Bases de Datos
Cáceres, 4 de septiembre de 2019

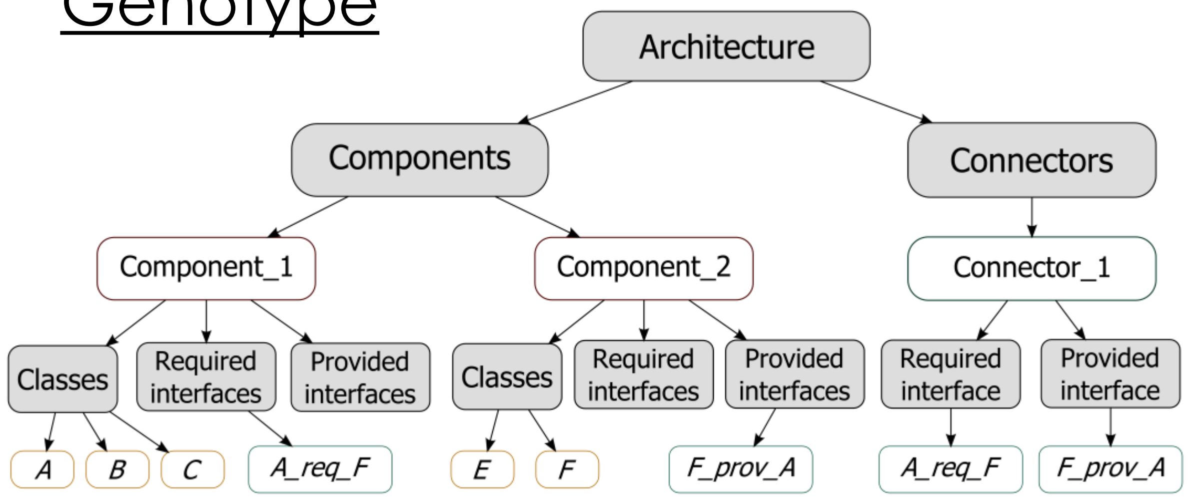
Interactive algorithm
Problem representation and constraints

Architectural solutions (individuals) *are **coded as multi-layered trees***

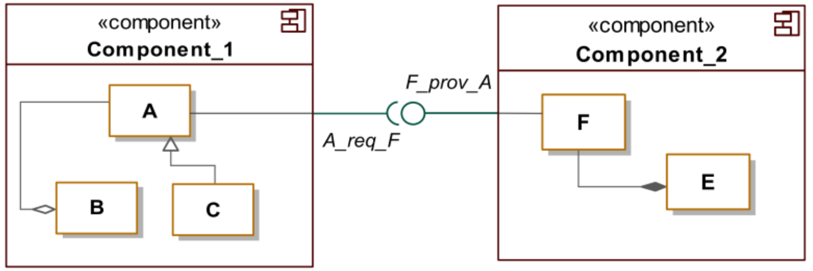
Initialization and constraints

1. Random distribution of classes
   - ✓ No empty components and no replicated classes
2. Set interfaces and connectors
   - × Isolated or mutually dependant components

Genotype



Phenotype



Interactive algorithm

Architectural preferences

1) Best component

$$pref_{bc} = max\{J(classes(c), classes(c^+))\}\ \forall c \in [1, n]$$

$$J(A, B) = |A \cap B| / |A \cup B|$$

2) Worst component

$$pref_{wc} = max\{1 - J(classes(c), classes(c^-))\}\ \forall c \in [1, n]$$

3) Best provided interface

$$pref_{bi} = max\{J(operations(interface(c)), operations(p^+))\}\ \forall c \in [1, n]$$

4) Worst provided interface

$$pref_{wi} = max\{1 - J(operations(interface(c)), operations(p^-))\}\ \forall c \in [1, n]$$

5) Number of components

$$pref_{nc} = \begin{cases} (n - n_{min})/(n^+ - n_{min}) & \text{if } n < n^+ \\ 1 - ((n - n^+)/(n_{max} - n)) & \text{if } n \geq n^+ \end{cases}$$

6) Metric in range

$$pref_{mr} = \begin{cases} 0 & \text{if } m^s < m_{min} \\ 1 - (m^s - m_{mid})/m_{mid} & \text{if } m^s \in [m_{min}, m_{max}] \\ 0 & \text{if } m^s > m_{max} \end{cases}$$

$$m_{mid} = (m_{max} - m_{min})/2$$

7) Aspiration levels

$$pref_{al} = \begin{cases} 1 & \text{if } ASF \leq 0 \\ 1 - ASF & \text{if } ASF > 0 \end{cases}$$

$$ASF = max\{w_k \cdot (f_k^s - z_k^*)\}$$