A novel algorithm for assessing services interfaces integrability

M. Crasso C. Mateos A. Zunino M. Campo

ISISTAN
Instituto de Sistemas Tandil

Universidad Nacional del Centro de la Provincia de Buenos Aires

CONICET

JAIIO - Jornadas Argentinas de Informática e Investigación Operativa
ASSE - Simposio Argentino de Ingeniería de Software
August 30-31 2012

Outline

Integration

Definition

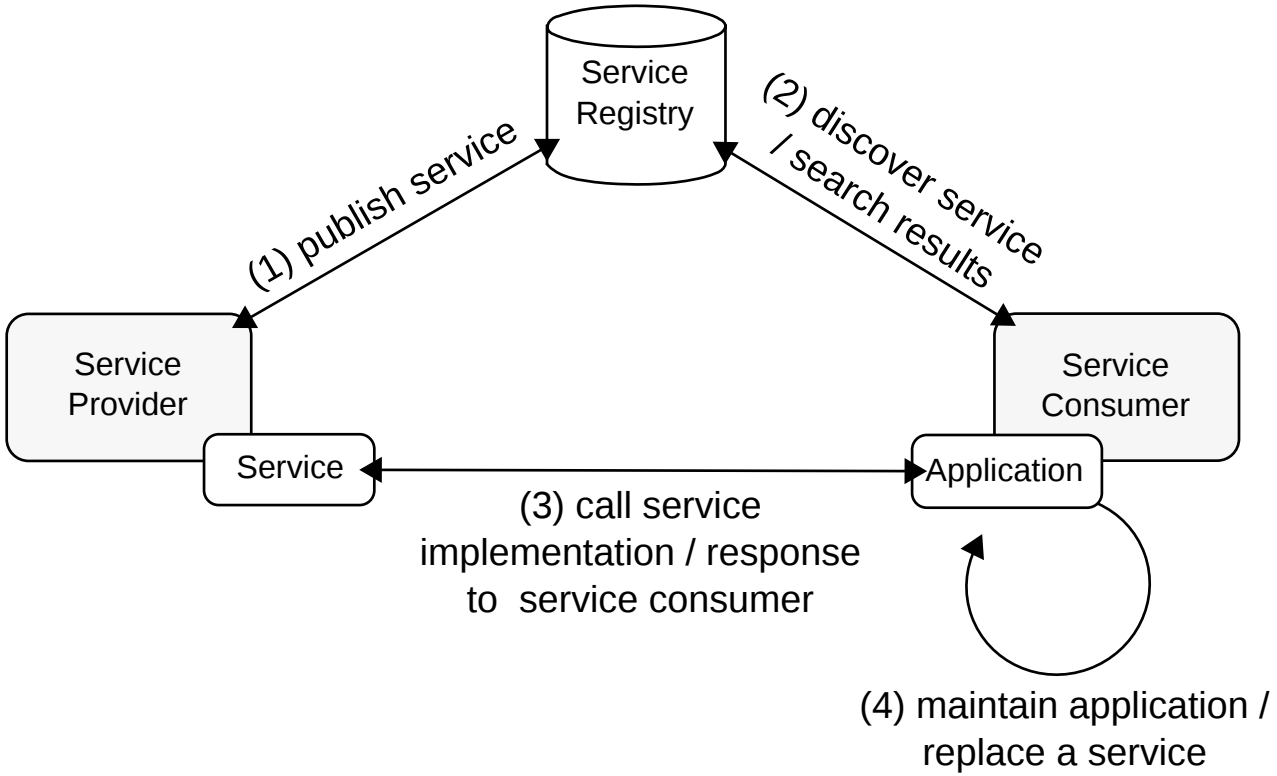
IEEE defines integration as “the process of combining software components, hardware components, or both into an overall system” [1]

Services Integration

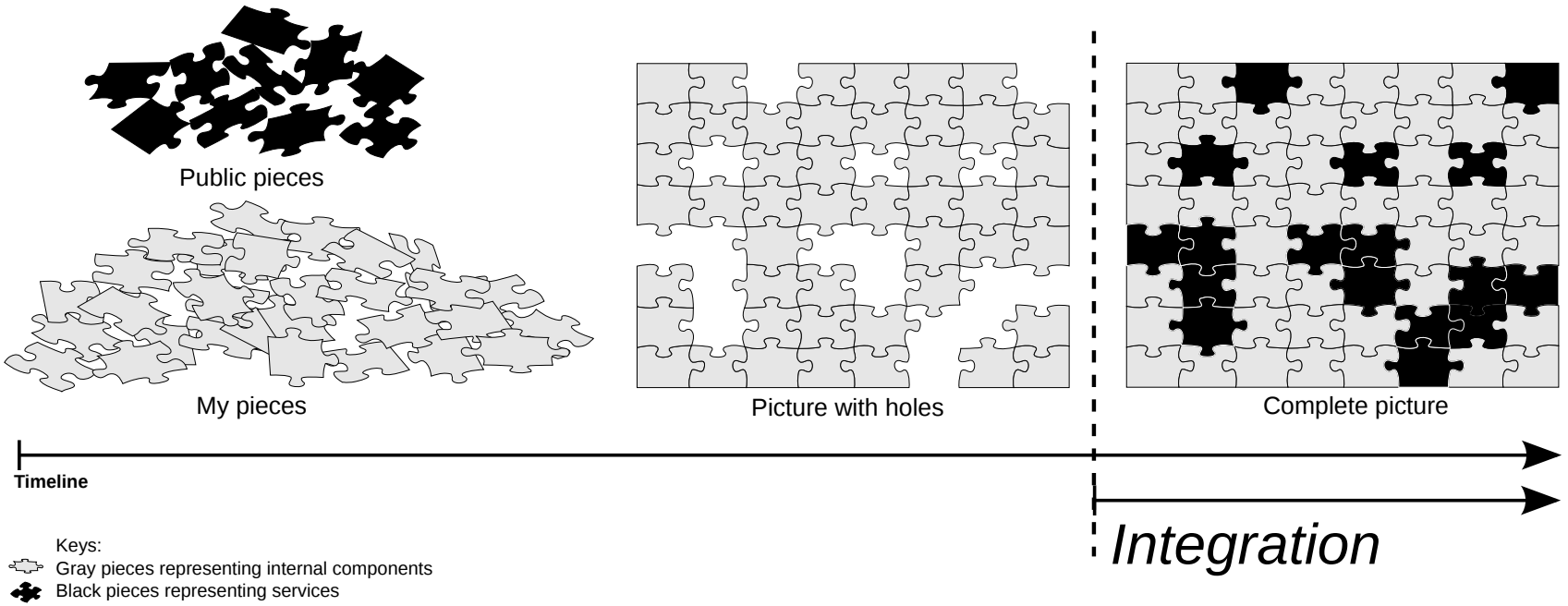
- The concept of software integration is essential for both already established computing paradigms and emerging ones:
  - Component-based software.
  - Service-Oriented Computing, and its satellite paradigms.
- Current main and preferred trends in software development are the integration of prefabricated software components, called ***services***.

Service-Oriented Computing

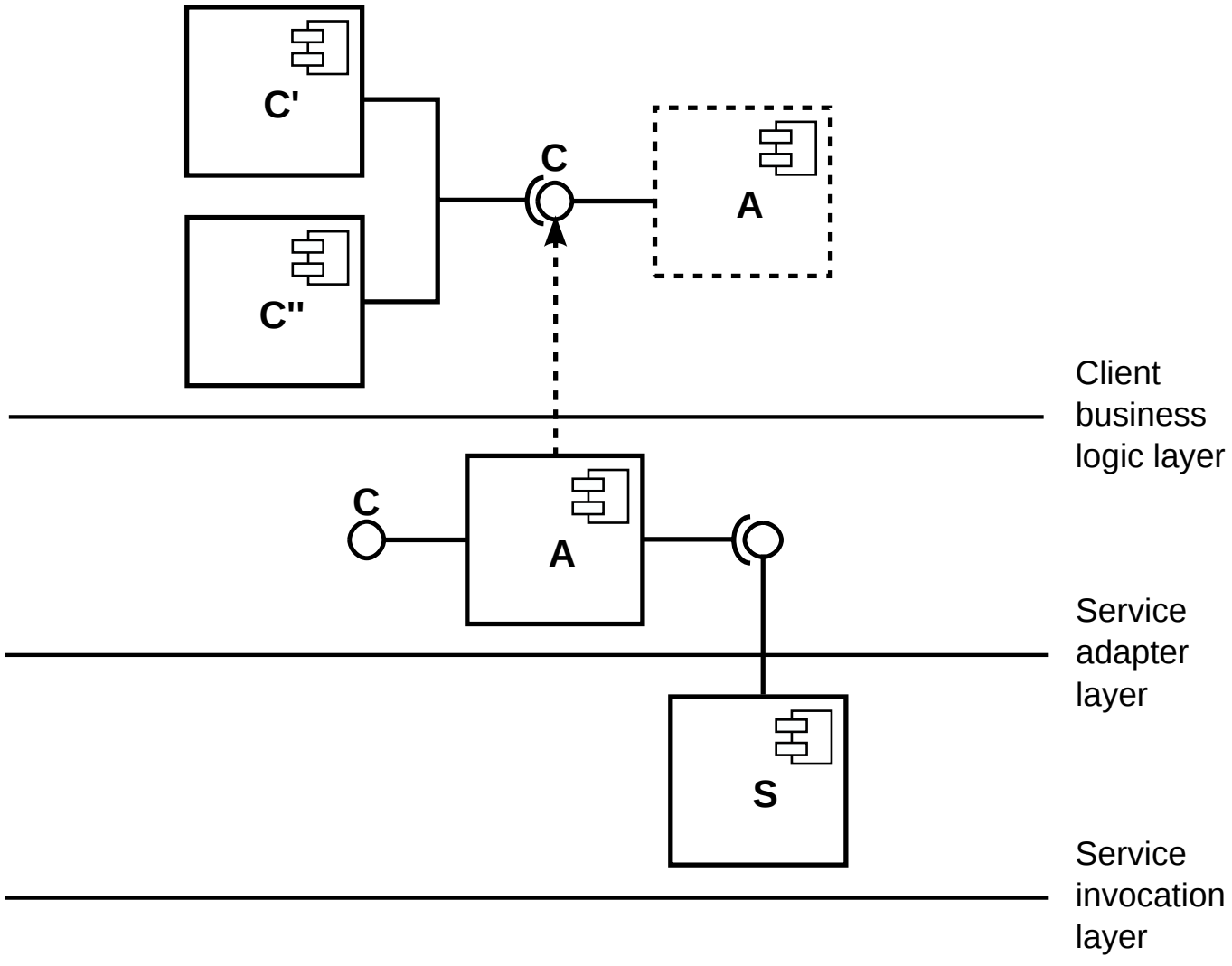
- SOC is a relative new and exciting paradigm for distributed computing



Conceptual Anatomy of a Service-Oriented Application



Maintainable Model of Service-Oriented Applications [3]



Services Integration Challenges

- Usually an interface (a set of operations with inputs-outputs) is the only available description of a service.
- There may be many services offering the same functionality.
- Differences between interfaces C and S impact on the adaptation layer.

Question

How we can determine which is the more integrable candidate?

Intuitive Criteria for Assessing Integrability

- The kind and amount of types conversions performed within the adaptation layer.
- The difference on the number of operations provided by each interface.
- The number of matched operations and the number of non-matched ones.
- The relevance of non-matched operations in terms of how many internal components call them.
- The degree of functional similitude between the operations of C and S.

Algorithm for Assessing Service Interface Integrability (AASII)

Inputs

- C: the interface, with n_C operations, expected by the application components.
- S: represents a candidate service interface with n_S operations.
- $\vec{r}$: relevance of each operation offered by interface C.

Output

- $[0, 1]$: a number representing how well the interface S integrates to those components that are already integrated to interface C.

Main Steps of AASII

1. calculate the normalized operations similitude matrix,
   1. e.g. by using any function from [4] provided it satisfies that
      $\forall i, i \epsilon [1, n_c] \wedge \forall j, j \epsilon [1, n_s] : f(op_i, op_j) \geq 0$

   $M = n_c \times n_s$

   $M_{i,j} = f(op_{C,i}, op_{C,j})$

   $\mu M_{i,j} = \frac{M_{i,j}}{\sum_{c=1}^{n_c} \sum_{s=1}^{n_s} M_{c,s}}$

2. calculate the best possible assignation between C operations and S operations,
   1. e.g. by using the Hungarian algorithm [5]

   $\vec{a} =< a_1, ..., a_{n_c} >$

   $\forall i, a_i \epsilon [0, n_s]$

   A : *assignations*

3. calculate the Integrability Metric

   $IM(\vec{r}, \mu M, \vec{a}, n_c, n_s) : [0, 1]$

Integrability Metric (IM)

$$IM(\vec{r}, \mu M, \vec{a}, n_c, n_s) = \frac{1}{\alpha} \times (\sum_{i=1}^{n_c} sim(op_i)) \times \frac{\sum_{i=1}^{n_c} rel(op_i)}{\sum_{i=1}^{n_c} \vec{r_i}} \times \frac{1}{\beta} \times [1 - (\frac{n_s - A}{n_s})]$$

$$sim(op_i) = \begin{cases} 0 & \vec{a_i} = 0 \\ \mu M_{i,\vec{a_i}} & \vec{a_i} \neq 0 \end{cases}$$

$$rel(op_i) = \begin{cases} 0 & \vec{a_i} = 0 \\ \vec{r_i} & \vec{a_i} \neq 0 \end{cases}$$

Theoretical Validation

- Researchers defined mathematical properties to which a software engineering metric should adhere:
  - Weyuker's work titled *"Evaluating software complexity measures" [6].*
  - Basili et at. work titled *"Property-based software engineering measurement" [2].*

- The proposed IM adheres to the following properties:
  - *Null Value* [2].
  - *Monotonicity* [2].
  - *Not Additive* [2].
  - *Nonnegativity* [6].
  - *Not Commutable* [6].

Algorithm Complexity

$$O(max(n_c \times n_s, f, n_c^3))$$

$O(n_c \times n_s)$: matrix calculation
$O(f)$: similitude function complexity
$O(n_c^3)$: Hungarian algorithm complexity

First Example (Scenario I: $n_c = n_s$)

- $n_C = n_S = 2$.
- $\vec{r} =< 1,1 >$, $\alpha = 1, \beta = 1$.
- $\mu M =$

	$op_{s,1}$	$op_{s,2}$
$op_{c,1}$	0.21	0.07
$op_{c,2}$	0.57	0.14

  $\vec{a} =< 2,1 >$.
- $IM(C,S) = (0.07 + 0.57) \times (2/2) \times 1 - [(2-2)/2] = 0.64 \times 1 \times 1 = 0.64$.

Second Example (Scenario II: $n_c < n_s$)

- $n_C = 3, n_S = 2.$
- $\vec{r} =< 1, 1, 1 >, \alpha = 1, \beta = 1.$
- $\mu M =$

μM	$op_{s,1}$	$op_{s,2}$	$op_{s,3}$
$op_{c,1}$	0.11	0.03	0.15
$op_{c,2}$	0.3	0.07	0.29

$\vec{a} =< 3, 1 >.$

- $(0.15 + 0.3) \times (2/2) \times 1 - [(3-2)/3] = 0.45 \times 1 \times (0.66) = 0.29.$
- $IM(C, S') < IM(C, S)$, because of the “dangling” operation.

Contributions

- A novel algorithm for assessing the effort needed to integrate a service into an application.
- The Integrability Metric (IM), a new software engineering metric.
- This represents a step towards alleviating the selection and integration of external services into service-oriented applications.

Future Work I

- To integrate the IM into the approach to discover services known as Query-By-Example (QBE).
- Conduct empirical evaluations of the proposed algorithm.

There are more details in the paper

- The theoretical validation.
- Four integrability scenarios.
- Related work discussion.

References I

[1] IEEE standard glossary of software engineering terminology.
IEEE Std 610.12-1990, page 1, 1990.

[2] L.C. Briand, S. Morasca, and V.R. Basili.
Property-based software engineering measurement.
Software Engineering, IEEE Transactions on, 22(1):68 –86, jan 1996.

[3] Marco Crasso, Cristian Mateos, Alejandro Zunino, and Marcelo Campo.
Easysoc: Making Web Service outsourcing easier.
Information Sciences, to appear, 2010.

References II

[4] Marco Crasso, Alejandro Zunino, and Marcelo Campo.
A survey of approaches to Web Service discovery in Service-Oriented Architectures.
Journal of Database Management, 22(1):103–134, 2011.

[5] Zvi Galil.
Efficient algorithms for finding maximum matching in graphs.
ACM Computing Surveys, 18:23–38, March 1986.

[6] E.J. Weyuker.
Evaluating software complexity measures.
Software Engineering, IEEE Transactions on, 14(9):1357–1365, sep 1988.

Questions...

Thank you!