The Data-Context-Interactions Paradigm Implementation In Morpheus

Zbyněk Šlajchrt1

1 Department of Information Technologies, Faculty of Informatics and Statistics,

University of Economics, Prague

W. Churchill Sq. 4, 130 67 Prague 3

zslajchrt@gmail.com

**Abstract:** Insert abstract in English here. Please follow the instructions for authors and respect maximum abstract length, which it should be between 100 – 200 words in length. Do not cite references to literature in the abstract.

**Keywords:** Max seven keywords or phrases state here, separated by commas.

# Introduction

Data, context, interactions (DCI) is a software paradigm introduced by Reenskaug (2008) and further elaborated by Coplien at al. (2010). Its main goal is to bring the end user's mental models and computer program models closer together. In other words, the user must feel that he or she directly manipulates the objects in computer memory that correspond to the images in his or her head. In this respect, DCI can be seen as a further development of the Model-View-Controller design pattern, whose goal is to provide the illusion of a direct connection from the end user brain to the computer “brain”, i.e. its memory and processor (Reenskaug & Coplien, 2009).

According to this paradigm, data, context and interactions are the three fundamental facets of the end user's interpretation of computer data. Each of these facets is considered a component of the application under development.

The data are represented by domain objects, which know everything about their state and how to maintain it, but know nothing about other objects in the system (Hasso et al., 2014). Coplien et al. (2010) use term *dumb objects* to refer to those domain objects. The technique for the identification of domain objects responsibilities is often called “Do it Myself” and is described in (Wirfs-Brock et al., 2003) and (Coad et al., 1997).

Next to the domain objects there is another kind of objects called interaction objects. These objects capture the interactions between domain objects as system functionality. Importantly, the interaction objects are treated by DCI as first class citizens. In contrast to domain objects the concept of interaction objects is a novelty introduced by DCI.

Only after the data are put into a context, in which they are subjects to interactions between them, the data can yield some information (Reenskaug, 2008). The context can be seen as the locus of use case enactment in the architecture encapsulating the interactions (or the roles) that define system dynamics (Coplien & Reenskaug, 2012).

In summary, in DCI, every use case is represented by means of the context. The context defines roles performing interactions between themselves. Each role in the context is played by one corresponding domain object (data, entity). The role contains the code that would otherwise reside in the object's class. Thus, roles effectively separate the stable part of the code from the unstable.

The context itself only defines roles and triggers the use-case. The Context and roles should reside in one dedicated file so that one could easily investigate the interactions.

DCI is anchored in object-oriented programming (OOP), however it must cope with some inherent flaws in OOP. It is generally accepted that OOP is very good at capturing the system's state by means of classes and their properties. OOP is also good at expressing operations with the state captured by a class, unless these operations involve some kind of collaboration with instances of other classes.

Nevertheless, implementing applications within the DCI paradigm is quite difficult in the traditional OOP, by large on account of the fact that OOP fails to express collaborations between objects (i.e. use cases) well, as shown in (Savkin, 2012). Let us consider an object that appears in several use cases and behaves very differently in each use case. Because of the lack of another concept in OOP a developer is forced to express such a use-case-specific behavior of the object as an operation in the object’s class. Unfortunately, it has several undesirable consequences:

First, there is no single file or other artifact dedicated solely to one use case, where a developer could see all interactions between objects. It makes the orientation in the code and its maintenance quite difficult.

Second, the whole behavior of the use case is scattered across the classes of the collaborating objects. It leads the developer to add a number of unrelated methods to classes with every new use case (causing higher coupling and lower cohesion of classes).

Third, it is virtually impossible to separate the stable part of the code, i.e. that which captures the data, from the variable part, i.e. the use cases (behavioral part).

As described in the “Related works” section, there are a number of DCI implementations in various programming languages, which more or less completely implement the DCI elements. In contrast to those implementations, which are built on top of the traditional object-oriented paradigm, the DCI demonstration shown in this paper is developed using Morpheus (Šlajchrt, 2015a), which is a proof-of-concept implementation of the *object-morphology* paradigm, a novel object-oriented conceptual framework for modeling mutable phenomena (Šlajchrt, 2015b).

# Related works

This section is divided into two sub-sections. The first subsection presents a number of concepts, which share some conceptual components with DCI. On the other hand, the second section deals with the DCI implementations in various languages.

## Concepts related to DCI

*Mixins* or *dynamic traits* provide a means for encapsulating a specific object behavior, such as interactions with other objects, into a special language construct reminding a class with no state (Groovy, 2016). An object may be turned into a DCI role by injecting a role mixin into it at runtime. However, it may be quite difficult to consistently combine multiple mixins at the level of a use case (Wikipedia, 2016).

*Dependency injection* allows segregating some object’s behavior into an external object. This segregation is accompanied by the specification of the interface between the original and external object. The behavior of the original object may be adapted to a given context (use case) by injecting an external object (role) at runtime. However, this approach suffers from the problem of *self-schizophrenia*, which is a condition in which the execution of some logic is carried out in the context of two or more objects (more selves), instead of only one. Such a condition may lead to subtle problems in the design (Sekharaiah, 2002). The problem of self-schizophrenia is treated well by DCI.

Aspect oriented programming …

Role-oriented programming …

Subject-oriented programming …

Other works are listed and commented in (Wikipedia, 2016).

## DCI implementations

… See: <http://fulloo.info/doku.php?id=existing_dci_implementations#scala>

Reenskaug, 2008: <http://folk.uio.no/trygver/2008/commonsense.pdf>

Reenskaug & Coplien, 2009: <http://www.artima.com/articles/dci_vision.html>

Coplien & Reenskaug, 2012, http://delivery.acm.org.zdroje.vse.cz/10.1145/2390000/2384782/p227-coplien.pdf?ip=146.102.19.70&id=2384782&acc=ACTIVE%20SERVICE&key=D6C3EEB3AD96C931%2EF07030D8BB94E8BC%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=764756558&CFTOKEN=14377343&\_\_acm\_\_=1458905528\_8b2b60ee7a437342e19692d46f1bd1ec

Hasso et al., 2014, http://search.proquest.com.zdroje.vse.cz/docview/1511434158?pq-origsite=summon

14] J. Coplien and G. Bjørnvig, "Lean Architecture: For Agile Software Development," 1st Edition, Wiley, West Sussex, 2010.

[15] R. Wirfs-Brock and A. McKean, "Object Design: Roles, Responsibilities, and Collaborations," Addison-Wesley, Boston, 2003.

[16] P. Coad, D. North and M. Mayfield, "Object Models: Strategies, Patterns, and Applications," Yourdon Press, Upper Saddle River, 1997.

Savkin, 2012, <http://www.sitepoint.com/dci-the-evolution-of-the-object-oriented-paradigm/>

Šlajchrt, 2015a – Morpheus

Šlajchrt, 2016b – thesis

Groovy Documentation. (2016). *Traits*. Retrived from: <http://docs.groovy-lang.org/latest/html/documentation/core-traits.html>.

Wikipedia, 2016, <https://en.wikipedia.org/wiki/Data,_context_and_interaction#cite_ref-20>

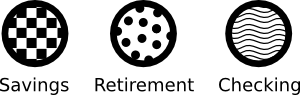
Sekharaiah K. Ch., Ram D. J., 2002, Object Schizophrenia Problem in Modeling Is-role-of Inheritance. [online]. http://users.jyu.fi/~sakkinen/inhws/papers/Sekharaiah.pdf.

# Research method

## Money transfer example

The use case scenario is this: the end user uses the bank terminal to transfer money from one account to another. He or she selects the source and the destination accounts from the list of accounts. Then he or she specifies the amount of money to be transferred and starts the transaction. Some exceptions can be raised, of course, for instance as long as there is not enough balance in the source account to perform the transfer.

For the sake of simplicity, let us assume that the data model of the bank application is just the list of the end user's accounts. Every account is represented by an object encapsulating some basic properties like the balance along with some basic operations like increaseBalance and decreaseBalance. We can expect that such a data model will be pretty alligned with the end user's mental model.



*Bank Accounts*

When transferring money the user will intuitively be familiar with the basic steps of the procedure. He or she will know that it is a simple interaction between two accounts, one playing the role of the source account and the other playing the destination account role and that the balance of the source account will be decreased by the amount of the transfered money while the destination's balance will be increased by the same amount.



*Transfer Money Use-Case*

It is important to note that the roles, i.e. the triangle and the rectangle have no identity. They can be regarded as costumes. It is the object playing the role that carries the identity.

In many object oriented programming languages it is usually pretty easy to express both the data and the abstract context (i.e. using roles that are not bound to the objects yet). What is not that easy, however, is the binding of objects to their respective roles in the context. In this use case the goal is to fill the holes in the rectangle and the triangle by the chosen account objects.



*Chosen Bank Accounts*



*Money Transfer*

This is the moment at which the things are becoming complex. Without modern programming concepts like mixins, traits, aspects or meta-programming we would hardly overcome this point. And yet any of these techniques has its own issues and does not perfectly fit to DCI.

See [this article on Wikipedia](https://en.wikipedia.org/wiki/Data,_context_and_interaction) dealing with various issues, or go [here to learn what *self schizophrenia*](https://en.wikipedia.org/wiki/Schizophrenia_(object-oriented_programming)) is.

So let us suppose we have managed to get through this difficult step and the objects are assigned to their roles. Now the context is ready to transfer the money. The user presses the **Go** button emitting a command that is delegated to the context, which will enact the use case.

## Using Morpheus

# Results

Static/dynamic hybrid, allows using Scala to implement DCI paradigm, in fact a dynamic Cake pattern

# Conclusion

# General information and styles

**This template is intended only for the contribution written in English! For Czech or Slovak written contribution, use the appropriate template.** Please use the following predefined formatting styles (Heading 1, Heading 2, main text, etc.) in this template.

Correctly formatted article according to this template should look like this:

[**EXAMPLE IN PDF**](http://aip.vse.cz/download/aip-template-en-example.pdf)

Articles can be published in English, Czech and Slovak. Articles are accepted only through [the journal publishing system](http://aip.vse.cz/index.php/aip/login) (if you have any questions, please contact the editor). Author will be informed about the acceptance of the paper by e-mail. The following communication (regarding the review process) goes on by the e-mail.

The minimum extent for the acceptance of the contribution as an original scientific paper (**Peer-reviewed papers section**) to the peer-review procedure is 4 pages and maximum is 20 pages. If you are interested in publishing longer paper, you have to inform the editor first. For contributions specified to the **section Peer-reviewed papers** is recommended this structure:

* Introduction
* Literature review and research methods
* Solutions and Results
* Discussion
* Conclusion
* Acknowledgement
* References

Parts can be structured into multiple subsections.

**Contributions to the non-reviewed sections** (Book reviews, Miscelanea and Reflections) at the end of the journal must be in the range from 2 to 6 pages.

# Heading 1 for the title of sections

Normal for the text of the paper.

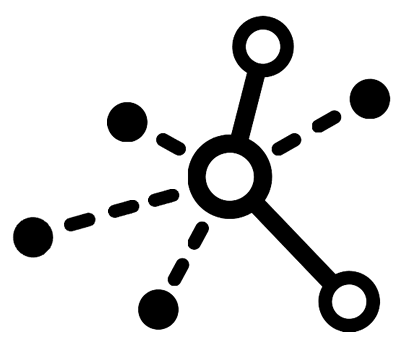
* Bullet A
* Bullet B
  + Bullet 1
  + Bullet 2
* Bullet C

## Heading 2 for the sub-titles of sections

### Heading 3

## Figure

All figures and tables must be centred and formatted according to the following examples.



**Fig. 1.** Caption for the captions of figures. Source (Berg, 2014, p. 57).

## Table

|  |  |  |
| --- | --- | --- |
| **AAA** | **BBB** | **CCC** |
| Aaa1 | Bbb1 | Ccc1 |
| Aaa2 | Bbb2 | Ccc2 |
| Aaa3 | Bbb3 | Ccc3 |
| Aaa4 | Bbb4 | Ccc4 |

**Tab. 1.** Caption for the captions of tables. Source (Berg, 2014).

## Source Code

#include <iostream>

using namespace std;

#define PI 3.14

int main ()

{

double r=4.0; // radius

double circle;

circle = 2 \* PI \* r;

cout << circle;

return 0;

}

## Equation

|  |  |
| --- | --- |
|  | **(1)** |

To create equations use the integrated editor "Equation editor" in MS Word. Equation insert in invisible table and each equation should be numbered.

# References

The contribution should primarily refer to the relevant scientific journals and conferences that are indexed in the Web of Knowledge and Scopus. References have to be **alphabetically sorted.** For links to references in the text, use the following examples using the author's surname and year of publication:

* **1 author** – Berg (2014, pp. 55-57) pointed out the problem… The problem is well-known (Magel, 2013a, p. 47) and further expanded in other papers (Lateg, 2013; Margel, 2013b; Apple, 2012).
* **2-3 authors** – Joergen a Jones (2009) improved used methods… Nevertheless in practice is used CUW method (Kang, Tucin & Kent, 2002).
* **More than 3 authors** – Skálová et al. (2010) provide the solution… The solution already exists (King et al., 2014).

**Examples of list of references**

To cite sources in the part of **References** use APA style (6th edition). The tutorial can be found here: <http://www.apastyle.org/learn/tutorials/basics-tutorial.aspx> In case of problems or doubts, we recommend to use a generator for APA citations on the internet. Below are examples of the most commonly used sources:

***Book:***

Berg, S. (2014). *Services marketing management*. London: Routledge.

Magel, J. (2013a) *Mastering data mining*. (1st ed.) Stockholm: Stockholm School of Economics.

Magel, J. (2013b) *Mastering GUHA. Stockholm*: Stockholm School of Economics.

***Article in journal:***

Joergen, P., & Jones, K. (2009). Random data analysis and measurement procedures. *Journal of Systems Integration*, 5(2), 55-85. Retrieved from <http://si-journal.org/joergen-jones-2009.pdf>

Kang, O., Tucin, J., & Kent, K. (2002). CUW methodics for marketing management. *Journal of Information Management*, 54(3), 1502-1535. doi: 10.7160/jim.2002.06784

***Article in other periodical (newspaper):***

Lateg, R. (2013, March 25). A survey of data provenance in e-science. *New York Review*, pp. 8-11.

***Conference contribution or chapter of the book:***

Skálová, U., Hopstal, H., Kuruc, T., & Krebs, W. (2010). The role of anomalous data in knowledge acquisition. In J. Jicinsky & P. Trejbal (Eds.), *5th International Conference on Informatics* (pp. 248-310). Kaunas: Walter Verlag.

King, U., Smith, L., Jones, E., Kain, W., & Wright, V. (2013). Power-law distributions in empirical data. In J. Rais (Ed.), *The design and analysis of spatial data structures* (1 ed., pp. 45-122). London: Research Life.

***Reference to the web page or e-document:***

Anders, Q. (2014, April 12). *Most-trusted brands*. Retrieved from <http://googleblog.blogspot.cz/anders-brands-2014>

Apple. (2012). *CFNetwork Programming Guide*. Retrieved from <https://develo-per.apple.com/library/mac/documentation/Networking/Conceptual/CFNetwork/CFNetwork.pdf>.