# Editorial

Hamid Mcheick

Département d'informatique et de mathématiques,
Université du Québec à Chicoutimi,
555 Boulevard de l'Université,
Chicoutimi Québec, G7H-2B1, Canada
E-mail: Hamid_Mcheick@uqac.ca

**Biographical notes:** Hamid Mcheick is currently an Associate Professor in the Computer Science Department of the University of Quebec at Chicoutimi (UQAC), Canada. He received his Master and PhD in Software Engineering and Distributed System from Montreal University, Canada. He is interested in software development and architecture for enterprise applications as well as in separation of concerns (component, services, aspect, etc.). His research is supported by many research grants he has received from the Canadian Government, University of Montreal, Centre de Recherche Informatique de Montreal (CRIM), University of UQAM, and University of UQAC.

The separation of concerns as a conceptual tool enables us to manage the complexity of the software systems that are developed and to delay as far as possible the binding of the various pieces or concerns related to a particular system. As such were the intent behind many approaches based on the divide and conquer concept: OORAM (Reenskaugh et al., 1995), use case approach (Jacobson et al., 1992), theme (Clark and Baniassad, 2004), building a foundation for structured requirements (Chernak, 2009) and many others. For example, identifying the different usage scenarios/use cases is the first step of separation in analysis time, in which unnecessary things are avoided. The OORAM methodology builds an application in terms of a combination of roles models, which are partial models describing specific collaborations between application entities (Reenskaugh et al., 1995). However, role models are combined before moving on to design and coding. When the idea is taken further to software packaging, better reuse, maintainability and interoperability can be achieved. Software packaging has different binding time: at analysis-design, at coding and finally run-time to adapt and compose concerns on the fly. Ideally, adding and composing concerns should involve conservative extension (what used to work will continue to work in an identical or a qualitatively equivalent fashion).

There have been a number of approaches aimed at modularising software around the natural boundaries of the various concerns, including subject-oriented programming (Harrison and Ossher, 1993), composition filters (Aksit et al., 1992), aspect-oriented programming (Kiczales et al., 1997), view-oriented programming (Mili et al., 2003, 1999), components-based system such as EJB, CORBA, .Net, OSGI (Wang and Qian, 2005) (OSGIAlliance, 2010), atomic and enhanced-modularity composition AEC (Msheik, 2010), service oriented architecture (Kumar et al., 2010; Brown, 2008), and many others. The growing body of experiences in using separation of concerns has identified a number of issues, both fundamental ones (what is an aspect and service, what is a concern, which concerns are separable, which services and aspects are compositional) as well as technical ones (how to use a particular technique to solve a particular problem). In general, researchers would study two issues:

- Focusing on the mechanics – and semantics – of aspect-oriented software development methods.
- Focusing on the semantics of separation of concerns. What is it that should be separated, and which concerns are even separable, before worrying about how to compose the artefacts that address them.

In summary, we need a framework which addresses the following questions:

a How can the software development process be improved based on components composition? One of the approaches has been developed is based on atomic and/or enhanced-modularity composition (Msheik, 2010). This approach proposes a model to shift and promote software construction from a traditional construction approach-based heavily on amalgamation of components and concerns to an approach relying increasingly on components composition.

b How can develop the concerns and keep them separate in the different process time ? In this respect, it is important to make the transition from requirements, analysis, design, implementation and run-time with separation of concerns in mind. For example, the transition from analysis to design consists of deriving an implementation of the functionalities specified at analysis time in a way that satisfies design-level constraints and addresses design-level concerns. Such concerns include error handling, synchronisation, logging, access to lower-level services, and the like. Addressing these concerns usually means adding code that crosscuts normal modularisation boundaries, i.e., typically objects and methods.

c How components and concerns can be composed and managed easily such as loosely coupling and lower development efforts are supported? Many approaches have been developed to compose these artefacts such as:

   1 subject oriented composition model (Ossher et al., 1995) based on composition rules

   2 building a foundation for structured requirements (Chernak, 2009) based on the composition of non-functional concerns with components, and many others.

In a software development process, quality attributes such as functionalities, reusability, maintainability, interoperability and usability should be satisfied in any development artefacts (concerns, components, etc.). These attributes can be realised using separation of concerns, service oriented computing and component-based approaches. In this special issue of *International Journal of Communication Networks and Distributed Systems (IJCNDS)*, the aim is to publish research contributions that significantly advance the state-of-the-art research in development through separation of concerns and service oriented computing to improve software quality.

## Acknowledgements

I would like to thank all the members of program committee for their helpful comments, suggestions and evaluations of the articles submitted to *IJCNDS* 2010, special issue – 'Separation of concerns and service oriented computing'. Sincere thanks especially to Professor Peter Kroft, Univerisity of Neuchâtel, Switzerland, Dr. Karim ElGuemhioui, Université du Québec en Outaouais, Canada, Dr. Haidar Safa, American University of Beirut (AUB), Lebanon, Dr. Hamdan Msheik, École de Technologie Superieure, Canada and Dr. Mohamed Dbouk, Université Libanaise (Faculté de science), for their assistance in producing this special issue. Many thanks to my wife, Fatima, for her ongoing patience and encouragement.

## References

Aksit, M., Bergmans, L. and Vural, L. (1992) 'An object-oriented language-database integration model: the composition filters approach', in *Proc. of ECOOP 92*, Springer-Verlag.

Brown, P.C. (2008) *Implementing SOA: Total Architecture Practice*, Addison-Wesley.

Chernak, Y. (2009) 'Building a foundation for structured requirements, aspect-oriented requirements engineering explained part 1', Better Software, available at http://www.StickyMinds.com.

Clark, S. and Baniassad, E. (2004) *Aspect-Oriented Analysis and Design: The THEME Approach*, Addison-Wesley/Pearson Education, Upper Saddle River, USA.

Harrison, W. and Ossher, H. (1993) ''Subject-oriented programming: a critique of pure objects',' in *Proc. of OOPSLA'93*, pp.411–428.

Jacobson, I., Christerson, M., Jonsson, P. and Overgaad, G. (1992) *Object-Oriented Engineering: A Use Case Driven Approach*, ACM Press, Addison-Wesley.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingetierand, J.M. and Irwin, J. (1997) 'Aspect-oriented programming', *Proceedings of ECOOP 97*, pp.220–242, CA, USA.

Kumar, B.V., Narayan, P. and Ng, T. (2010) *Implementing SOA Using Java™ EE*, Addison-Wesley.

Mili, H., Dargham, J., Bendaloul, S. and Mcheick, H. (1999) 'Separation of concerns and typing: a first stab', *Workshop MDSOC in the OOSPLA'99*, November, Colorado, USA.

Mili, H., Mcheick, H. and Sadou, S. (2003) 'CorbaViews: distributing objects that support several functional aspects', *Journal of Object Technology*, August, pp.209–227, Zurich, Suisse.

Msheik, H. (2010) 'Software construction by composition of components', These PhD, École de Technologie Supérieure, Montréal, Canada.

OSGIAlliance (2010) Available at http://www.osgi.org/About/WhyOSGi.

Ossher, H., Kaplan, M., Harrison, W., Katz, A. and Kruskal, V. (1995) 'Subject-oriented composition rules', in *Proceedings of OOPSLA'95*, 15–19 October, pp.235–250, Austin, TX, USA.

Reenskaugh, T., Wold, P. and Lehne, O.A. (1995) *In Working with Objects*, Prentice-Hall.

Wang, A.J.A. and Qian, K. (2005) *Component-Oriented Programming*, John Wiley & Sons.