Abstract:

The emergence of new distributed computing scenarios such as ad-hoc, mobile, peer-to-peer, and pervasive computing techniques motivate the need for novel approaches to support and facilitate the development and execution of complex software applications. Embedded hardware and software elements in these applications are resource-constrained and power-limited devices. Software development for these distributed software elements are often contextual and often strictly related to the environment in which they reside. These individual software elements need to exhibit simple, yet well-defined underlying patterns, thereby aiding the development of scalable, complex software systems capable of demonstrating domain-specific intelligence. Such heterogeneous (often in terms of work function), aggregated spatial distribution patterns of individual elements are almost universal across living organisms. In nature, such aggregations do seem to confer advantages to group members. Understanding and mimicking these social organizations, and aggregation behaviors such as selflessness, group living and self-organization, should significantly contribute to a simple, yet immensely scalable, software computational models. Currently, we do have some maturity of research to develop software-based mechanisms to not only identify and close the loop for continuous improvement, but also selectively communicate essential information at each resolutional layer to benefit “coordination” at higher nested levels. At each level, the issue then becomes “what is essential information” and “how much information is sufficient”? This helps formulate a reliable approach to define clear pseudo-global perspectives. However, researchers in theory and practice of software systems modeling have concentrated on the issues of design correctness, provability, etc. as parameters to determine larger “composability”. If we can couple those advances with effective “communication” and “coordination” mechanisms, several problems in integrating software systems will diminish and scalable, seamless software-driven infrastructures can emerge. Developing modeling methodologies that help to appropriately tune the quest for such essential information and analytically prove their sufficiency (in a domain specific sense) within a larger context is therefore, vitally important for successful large scale deployment.