Architecture, particularly software architecture, is currently a subject of significant resonance in the software engineering literature. There are conferences devoted to it, special issues of fashionable software engineering journals dedicated to it, working groups and workshops exploring it, and it can usually be found in the list of topic areas in Calls for Participation for conferences similar to this one.

What’s all the fuss about?

This talk will provide an overview of the current field of study known as software architecture, and point out where we are simply re-visiting old concepts under a new name, and where the field is making legitimately new progress. We will explore the uses of software architecture, including its utilization as each of the following:

- As a basis for stakeholder communication: Project team members, managers, and customers all turn to the architecture as the basis for understanding the system, its development, and how it works during execution.
- As a project blueprint: The choice of architectural components is institutionalized in the developing organization’s team structure, work assignments, management units, schedule and work breakdown structures, integration plans, test plans, and maintenance processes. Once it is made, an architectural decision has an extremely long lifetime and survives even outside of the software that it describes.
- As a blueprint for product line development. An architecture may be re-used on other systems for which it is appropriate. If managed carefully, an entire product family may be produced using a single architecture. In this case, the importance of an appropriate architecture is magnified across all the projects it will serve.
- As the first approach to achieving quality attributes: An architecture can either allow or preclude the achievement of most of a system’s targeted quality attributes. Modifiability, for example, depends extensively on the system’s modularization, which reflects the encapsulation strategies. Reusability of components depends on how strongly coupled they are with other components in the system. Performance depends largely upon the volume and complexity of the inter-component communication and coordination, especially if the components are physically distributed processes. Thus, an architecture embodies decisions about quality priorities and tradeoffs, and represents the earliest opportunity for evaluating those decisions and tradeoffs.

Finally, we will discuss current trends and future possibilities in bring automated support to bear in the creation, representation, evaluation, evolution, and extraction of software architecture in support of those goals.