Chapter 7
Conclusions

This chapter concludes the study in this thesis providing answers to the research questions, remarks about the research contribution, and some open issues for future research.

7.1 Answers to the Research Questions

In Chapter 1, we described the role of up-to-date architectural views in an architecture-centric evolution approach. With the Philips MRI scanner as a representative large and complex software-intensive system, we explained that architects have to produce up-to-date architectural views to, among other things, support the description and analysis of dependencies, and the description and analysis of the actual runtime structure and behavior of a software system. To this end, we formulated a set of research questions to investigate how to support architects in the construction and use of up-to-date architectural views. Below, we answer the formulated questions.

- **RQ-1.** *What is the state-of-the-art in dependency analysis and its value for the architecture-centric evolution of a large and complex software-intensive system?*

  To answer this research question, we conducted a systematic literature review about dependency analysis (Chapter 2). Through the review, we identified and selected relevant articles from the literature. As a result of this process, we characterized the state-of-the-art in dependency analysis with the existing definitions of dependency and the types of dependencies (see Section 2.4), the set of activities that dependency analysis solutions claim to support (see Section 2.5), and a classification of the existing dependency analysis solutions (see Section 2.6). These elements as a whole represent an overview, which conveys the existing support that practitioners and researchers can use, implement, or extend according to their context and needs.

  To identify the actual value of the state-of-the-art in dependency analysis, i.e. for the architecture-centric evolution of a large and complex software-intensive system, we conducted an assessment taking into account the practical characteristics of the software embedded in the Philips MRI scanner and its development process (see Section 2.7). The assessment is about the applicability of the identified definitions related to dependencies and the identified types of dependency analysis solutions. The result of the assessment shows that the state-of-the-art in dependency analysis proposes a wide range of solutions to support actual development activities. However, several key practical requirements are not properly addressed yet. Mainly, suitable solutions are required to manage, e.g. collect, abstract or aggregate, and present, the heterogeneous and massive amount of data that can be obtained from the source code, documentation, or runtime activity made available during the incremental development of large software-intensive systems. Hence, there are research opportunities that can have a strong impact on the way that software organizations develop large and complex software-intensive systems.

- **RQ-2.1.** *What are the useful runtime concepts that we can use to describe the actual runtime behavior and structure of the software embedded in a software-intensive system?*

  The metamodel presented in Section 3.3.2 and Section 5.3.3 organizes a set of runtime concepts that we found useful to describe the runtime structure and behavior of the software embedded in the Philips MRI scanner. We found these concepts by reviewing the literature and taking into account the preference or requirements of a group of architects and designers of the Philips MRI scanner. The metamodel links architectural concepts (e.g., execution scenario, task, software component, and relationships between them) to actual runtime platform elements (e.g., processes, threads, and their activity) and to other important elements or resources (e.g., data, code, and hardware).

  It is important to notice that the elements and relationships in the metamodel are not an exhaustive compilation of all possible useful concepts that can be used to describe the runtime of a software system. Instead, we present them as a representative set that can be customized or extended. For example Chapter 4 and Chapter 6 present case studies where we customized the metamodel to cope with the complexity and size of the system, and obtain detailed information.

- **RQ-2.2.** *How to obtain up-to-date information about the runtime of the software embedded in a software-intensive system?*

  In Section 3.4, we present a dynamic analysis technique, which outlines a set of activities and mechanisms that one can put in place to obtain up-to-date information about the runtime of a system like the Philips MRI scanner. With this technique, runtime information is obtained from sources like logging and runtime measurements applying a rule-based mechanism. The rule-base mechanism, described in Section 3.4.2, is to map data from logging or runtime measurements to instances of the elements and relationships organized by the execution metamodel, see Figure 3.3.

  In Table 3.2 and Section 4.3.2, we illustrate the sort of data and the corresponding runtime information that we obtained applying the dynamic analysis technique within the development of the Philips MRI scanner. In practice, the support provided by the dynamic analysis technique is characterized by how it can be used to extract up-to-date runtime information using existing infrastructure, e.g., logging, or other means, e.g. runtime measurements, that produce few or none overhead in the system's performance and the development process.

- **RQ-2.** *How to construct execution views of the software embedded in a large and complex software-intensive system?*

  The architecture reconstruction approach presented in Chapter 3 outlines how to construct execution views following an iterative and problem-driven process. The approach shows that a synergy between the execution metamodel, a set of viewpoints, the dynamic analysis technique, and the sources of runtime information is the key to execution views. The application of the approach consist of two phases, i.e., reconstruction design and reconstruction execution. An architect can follow these phases for the elicitation of the problem that triggers the construction of the execution view, and then, iteratively construct and present the required model(s) for an execution view.

  In Chapters 3, 4, and 6, we show that a set of representative execution scenarios for the problem that triggers the construction of the view, the sources of runtime information, e.g., logging and runtime measurements, some initial domain knowledge, and proper viewpoint play a fundamental role to construct and up-to-date and useful execution view. The main benefit of the architecture reconstruction approach is that it enables the construction of execution views with high-level information at first and then, if it is needed by the stakeholders, detailed information to zoom in on areas of special attention.

- **RQ-3.** *What are the useful viewpoints for execution views of a software-intensive system?*

  In Chapter 5, we present four viewpoints for execution views. These viewpoints proved to be useful guidelines to construct and use execution views through several development projects with Philips MRI scanner. The execution profile viewpoint guides the construction and use of views to describe and analyze the relations between the system functionality, system functional components, and actual runtime elements. The execution deployment viewpoint guides the construction and use of views to describe and analyze the allocation of system execution elements to processing nodes and the environment into which the system is deployed. The resource usage viewpoint guides the construction and use of views to describe and analyze the aspects that govern how the software actually accesses and use at runtime data, code, and mainly hardware elements, e.g., memory and processors. The execution concurrency viewpoint guides the construction and use of views to describe and analyze the actual control flow and data flow between runtime elements.

  The value of the viewpoints has its roots in the systematic way that we defined and documented them. We devised the knowledge in the viewpoints customizing example viewpoints from the literature and the elicitation of the requirements of a representative development organization. The documentation of the viewpoints makes the knowledge about the construction and use of execution views reusable and implicit. The knowledge in every viewpoint conveys the stakeholders' concerns, the type of execution models that address such concerns, how to construct the models to build a view, and when and how to use the constructed execution views.

- **RQ-4** . *How is the construction and use of execution views embedded in the incremental development process of a software-intensive system?*

  Architectural descriptions like execution views need to be constructed and used multiple times during the incremental development of a system like the Philips MRI scanner. The characteristics of the development process and the organization drive the application, i.e. use, reuse, and embed, of architecture reconstruction solutions. In Chapter 6, we report a top-down strategy to facilitate the systematic construction and use of execution views during the incremental development process taking into account the aforementioned characteristics. The strategy shows that the systematic construction and use of execution views is a successive set of iterations with the architecture reconstruction approach in combination with analysis activities.

  The iterations with the architecture reconstruction approach and analysis activities go from the construction of a high-level hypothesis about the runtime, to the construction of up-to-date models with very detailed information. The analysis of high-level hypothesis helps the architect to collect background information and define the scope of the problem to be addressed. The analysis of the execution models with detailed information help the developers to find the root cause of the issue related to the problem at hand and the design and implementation of a solution to fix it.

- **RQ-5** . *What is the actual contribution of constructing and using execution views in the incremental development of a software-intensive system?*

  Constructing and using execution views increases the ability to respond effectively to change, in particular to fit in changing requirements that influence on the runtime of the system. For example, in Chapter 6, we report that construction and use of an execution view for the start-up process of the Philips MRI scanner. The execution models in the view enabled the identification of issues in this representative runtime feature, as well as the analysis of solutions. As a result, the development organization were able to quickly reduce about 30% the start-up time of the scanner, and the setting of a system benchmark to monitor the system's performance in future evolution steps.

  In a wider scope, the construction and use of execution views contributes to general development activities. As part of the research conducted to define the viewpoints for execution view, we identified that system understanding, project planning, communication, and conformance of design and implementation are often supported by execution views (see Section 5.3.3). For example, execution views are used for system-specific education and training of new developers, who need to create mental models of the overall system, the system components they develop, and their relations (dependencies) with the rest of the system components. In Chapter 3 and Chapter 4, we provide more concrete examples of how execution views contribute to other development activities, including dependency analysis, feature analysis, and analysis of alternative designs.

7.2 Research Contribution and Extrapolation

The body of knowledge of this thesis has been reviewed, presented and it even obtained awards in conferences and journals of two distinct scientific communities: reverse engineering (Callo Arias et al. 2008, Callo Arias, America and Avgeriou 2009a) and software architecture (Callo Arias, America and Avgeriou 2009b). In addition, the new version of the ISO/IEC 42010 Std. Recommended Practice for Architectural Description of Software-Intensive Systems, cites the early definition of the execution viewpoint catalog (Callo Arias, America and Avgeriou 2009b) as a representative example of how to define viewpoints to describe the architecture of software intensive-systems. In overall, the research presented in this thesis contributed to the state of practice in Philips Healthcare MRI and to the state of the art in the architecture and reverse engineering fields.

The combination of the research results presented in this thesis build up a pragmatic solution for reverse architecting the runtime structure and behavior of a software-intensive system like the Philips MRI scanner. The main characteristics of the solution are:

- The solution is an iterative process, rather than a single tool solution, that addresses important practical needs imposed by the incremental development of an existing software-intensive systems (see Section 3.7).
- The solution uses resources that are available as part of the system infrastructure, i.e. logging, or sources that can be easily collected using system platform tools, i.e., runtime measurements (see Section 3.4).
- The solution is supported by explicit and well-documented guidelines, i.e, execution viewpoints, that can be reused across development projects (see Section 5.5).
- The solution enables the construction of execution views, which practitioners found useful to support a number of usual activities, e.g. feature analysis, dependency analysis, conformance and realization analysis (see Section 3.5), and performance analysis of data- and computation-intensive features (see Section 6.4 and 4.5).

In practice, the support provided by the solutions is required even for well-structured systems. We consider that the knowledge presented in Chapters 2 and 3 can be applied or extended by other architect to determinate dependencies in the runtime structure and behavior of similar or more complex software-intensive systems like the Philips MRI scanner. Similarly, the knowledge presented in Chapter 4 and 6, can be applied, customized, and extended by architects that need to fit in new technology, redesign data- or computation-intensive features, or verifies the system performance.

7.3 Open Issues and Future Research

Following an industry-as-laboratory research approach in a development organization like Philips Healthcare MRI gave us the chance to work in the context that software engineering research aims at studying and improving. Therefore, we experienced the benefits and challenges that this context provides to conduct quality and rigorous research, respectively. As we described in Section 1.6.1, the benefits includes investigating relevant problems and their root causes, propose and develop solutions, and try out and validate the solutions in a real context. However, each of these benefits posed challenges that influenced the quality and rigor of our research. To identify relevant problems, we invested time studying related literature and moreover joining and observing the development process. when we were able to identify a relevant problem, the next challenge was to find the right problem owner. Without a problem owner, a problem becomes irrelevant and trying out or validating a given solution is not beneficial and objective.

Philips Healthcare MRI, as a development and business organization, is a hub of activity where project goals, staff, and priorities change continuously. Finding and keeping a problem owner in this environment is not easy. We needed to search for practitioners and convince them that their current way of working could be improved and still fulfil the goal and priorities of the organization. In practice, this means that our time and resources were exclusively dedicated to research activities (including training, meetings, presentations, informal and informal interviews, chats at the coffee corner, tests, and demos) within Philips Healthcare MRI. This exclusive dedication and the nature of the research project leave as an open issue, the replication and validation of our contribution with more than one development organization or software-intensive system.

The scalability of the execution metamodel, presented in Section 3.3.2, is an open issue that future research can address. We designed the metamodel taking into account the architecture and implementation technology of the Philips MRI scanner. For future research, it will be interesting to know how the metamodel matches systems with different implementation technology and architecture. This future work will enable the identification of variations, extensions, and other customizations for the metamodel, as wells as, the construction of a catalog of execution metamodels. The execution viewpoints, defined and documents in Chapter 5, frame four sets of concerns related to the runtime of software intensive system. Another open issue is to identify whether these sets of concerns cover all the relevant runtime related concerns. Consequently, future research should investigate what are the other runtime related concerns that execution viewpoints should frame. The resulting contribution may imply the inclusion of more execution viewpoints, or the fine-tuning of some of the current execution viewpoints.

The experience and feedback of fellow researchers and practitioners can contribute in a number of aspects to learn more about the challenges and benefits of constructing and using execution views. Future research can address how to design and implement techniques and tools to trace and abstract runtime data. The contribution of this future research can help us to improve and make available infrastructures that assure the availability of information to construct execution views. Another open issue is the design and implementation of dedicated and specialized tool-support to construct and present execution views. Dedicated tool support may speed-up the construction of execution views, and the implementation and reuse of the execution viewpoints' guidelines.