Safety-critical applications have to function correctly and deliver high level of quality-of-service even in the presence of faults. This thesis deals with techniques for tolerating effects of transient and intermittent faults. Re-execution, software replication, and rollback recovery with checkpointing are used to provide the required level of fault tolerance at the software level. Hardening is used to increase the reliability of hardware components. These techniques are considered in the context of distributed real-time systems with static and quasi-static scheduling.

Many safety-critical applications have also strict time and cost constrains, which means that not only faults have to be tolerated but also the constraints should be satisfied. Hence, efficient system design approaches with careful consideration of fault tolerance are required. This thesis proposes several design optimization strategies and scheduling techniques that take fault tolerance into account. The design optimization tasks addressed include, among others, process mapping, fault tolerance policy assignment, checkpoint distribution, and trading-off between hardware hardening and software re-execution. Particular optimization approaches are also proposed to consider debugability requirements of fault-tolerant applications. Finally, quality-of-service aspects have been addressed in the thesis for fault-tolerant embedded systems with soft and hard timing constraints.

The proposed scheduling and design optimization strategies have been thoroughly evaluated with extensive experiments. The experimental results show that considering fault tolerance during system-level design optimization is essential when designing cost-effective and high-quality fault-tolerant embedded systems.
Safety-critical applications have to function correctly and deliver high level of quality-of-service even in the presence of faults. This thesis deals with techniques for tolerating effects of transient and intermittent faults. Re-execution, software replication, and rollback recovery with checkpointing are used to provide the required level of fault tolerance at the software level. Hardening is used to increase the reliability of hardware components. These techniques are considered in the context of distributed real-time systems with static and quasi-static scheduling.

Many safety-critical applications have also strict time and cost constrains, which means that not only faults have to be tolerated but also the constraints should be satisfied. Hence, efficient system design approaches with careful consideration of fault tolerance are required. This thesis proposes several design optimization strategies and scheduling techniques that take fault tolerance into account. The design optimization tasks addressed include, among others, process mapping, fault tolerance policy assignment, checkpoint distribution, and trading-off between hardware hardening and software re-execution. Particular optimization approaches are also proposed to consider debugability requirements of fault-tolerant applications. Finally, quality-of-service aspects have been addressed in the thesis for fault-tolerant embedded systems with soft and hard timing constraints.

The proposed scheduling and design optimization strategies have been thoroughly evaluated with extensive experiments. The experimental results show that considering fault tolerance during system-level design optimization is essential when designing cost-effective and high-quality fault-tolerant embedded systems.