**Jens Ehlers: Self-Adaptive Performance Monitoring for Component-Based Software Systems**

**1. Gutachter:** Prof. Dr. Wilhelm Hasselbring (Universität Kiel)
**2. Gutachter:** Prof. Dr. Florian Matthes (Technische Universität München)
**Datum der Prüfung:** 20.04.2012

**Zusammenfassung:**
Effective monitoring of a software system's runtime behavior is necessary to evaluate the compliance of performance objectives. In addition to studying the construction and evolution of software systems, the software engineering discipline needs to emphasize the interest on robust and flexible software system operation, including means for continuous monitoring.

Though performance is a critical characteristic for software systems, tools addressing application performance monitoring, i.e. monitoring the operation of software systems at application level, are rarely used in practice. This prevalent negligence is expressed by the following symptoms: (1) a posteriori failure analysis, i.e. appropriate monitoring data is seldom collected and evaluated systematically before a failure or performance anomaly occurs, (2) inflexible instrumentation, i.e. probes are injected only at a limited number of fixed measuring points such that component recompilation and redeployment are required for future modifications, and (3) inability of tracing in distributed systems, i.e. tracing of user requests or transactions beyond the borders of components or their execution containers is not supported or not applied.

This thesis has emerged in the context of the Kieker monitoring framework, which targets the above shortcomings. A finding of our experimental overhead evaluation is that it is feasible to instrument probes at a multitude of possibly relevant measuring points, as long as not all of them are active at the same time during operation. Therefore, this thesis proposes a self-adaptive performance monitoring approach allowing for dynamic activation of probes and measuring points. As typical for autonomic systems, a control feedback cycle manages the adaptation of the monitoring coverage at runtime. The solution is based on OCL-based monitoring rules that refine the monitoring granularity for those components that show anomalous responsiveness.

The monitoring data includes performance measures such as throughput and response time statistics, the utilization of system resources, as well as the inter- and intra-component control flow. Based on this data, performance anomaly scores for each provided service and component-inherent operation are computed. The presented anomaly scoring algorithms are based on time series analysis and distribution clustering, respectively.

This self-adaptive performance monitoring approach for component-based software systems reduces the business-critical failure diagnosis time, as it saves time-consuming manual debugging activities. The approach and its underlying anomaly scores are extensively evaluated in lab experiments, e.g. using the SPECjEnterprise2010 industry benchmark. The evaluation results, in combination with the implementation of the Kieker tool, demonstrate the feasibility and the practicability of the approach.

Veröffentlicht als:
Jens Ehlers: Self-Adaptive Performance Monitoring for Component-Based Software Systems, Kiel Computer Science Series, 2012, (Books on Demand, ISBN 978-3-8448-1447-7). Online unter http://eprints.uni-kiel.de/15495/

**Stefan Gudenkauf: Domain-Specific Modelling for Coordination Engineering**

**1. Gutachter:** Prof. Dr. Wilhelm Hasselbring (Universität Kiel)
**2. Gutachter:** Prof. Dr. Michael Hanus (Universität Kiel)
**Datum der Prüfung:** 06.07.2012

**Zusammenfassung:**
Mit der Verbreitung von Multi-Core-Rechnern im Verbrauchermarkt und dem aktuellen Forschungsinteresse der Industrie an Many-Core-Rechnern haben parallele Systeme einen festen Platz im Massenmarkt eingenommen. Damit einhergehend werden die Anzahl paralleler Prozessierungseinheiten und Hochbandbreit-Kommunikation als die neuen Performance-Treiber angesehen. Dieser Umbruch wird jedoch nicht durch einen ähnlichen breitenwirksamen Umbruch in der Anwendungsprogrammierung mitgetragen. Multi- und Many-Core-Rechner erfordern jedoch, Nebenläufigkeit auch in der herkömmlichen Softwareentwicklung so weit wie möglich auszureizen, um weiterhin vom Hardware-Fortschritt profitieren zu können.

Hieraus ergeben sich drei Herausforderungen für die alltägliche Softwaretechnik: Portabilität und die menschliche Wahrnehmung erfordern die Bereitstellung standardisierter Nebenläufigkeitsmodelle höherer Abstraktionsebene, die Kontinuität mit gängigen und weit verbreiteten Technologien muss gewährleistet sein, und bestehende Muster und Praxiserfahrungen der Entwicklung parallel laufender Software muss an Software-Ingenieure aller Arten und Anwendungsgebiete verbreitet werden. Dies steht im Gegensatz zum High-Performance Computing, wo allein die maximale Leistungsausbeute im Vordergrund steht.

In der vorliegenden Arbeit verbinden wir Koordinationsmodelle mit Techniken der modellgetriebenen Softwareentwicklung (MDSD), um diesen Herausforderungen zu begegnen: Wir verwenden Space-Based Systems (SBS) als Koordinationsmodell, um die Interaktion von gleichzeitig ablaufenden Prozessen zu spezifizieren, und bilden es zur Spezifikation und als Mittel zur Verbreitung auf die Business Process Model and Notation (BPMN) 2.0 Spezifikation ab – ein weit verbreitetes Modell für die Definition von Geschäftsprozessen. Zudem verwenden wir Modell-Transformationen, um die Einhaltung des Koordinationsmodells im Programmcode durchzusetzen, und stellen Coordination Engineering als konkrete Methode und als Leitlinie für die Anwendung bereit.

Als Beleg der Machbarkeit haben wir das resultierende Koordinationsmodell Space-Coordinated Processes (SCOPE) in zwei Implementierungen ausgeführt: als Programmier-Bibliothek Process Coordination Library (PROCOL) und als domänenspezifische Sprache (DSL) mit zugehöriger Entwicklungsumgebung. Die Ergebnisse werden mit Proto-Benchmarks und Experimenten mit Anwendungen für massive Parallelität und für Point-Feature-Label-Placement (PFLP) untermauert.

Der Beitrag dieser Arbeit kann wie folgt zusammengefasst werden:

- SCOPE ist ein konzeptionelles Koordinationsmodell, das die SBS-basierte Choreographie unabhängiger Prozesse mit der prozessinternen Orchestrierung feingranularer Workflow-Aktivitäten verbindet.
- PROCOL ist eine Implementierung von SCOPE als Programmierbibliothek, um die Entwicklung nebenläufiger SBS-basierter Programme in der Programmiersprache Java zu vereinfachen. PROCOL kann als interne DSL für SCOPE verstanden werden.
- SCOPE DSL kann zusammen mit der zugehörigen Entwicklungsumgebung als externe textuelle DSL für SCOPE verstanden werden. Beide basieren auf dem Xtext Sprachframework und sind interoperabel mit beliebigen BPMN-konformen Werkzeugen.
- Coordination Engineering ist eine Methode, die das Koordinationsmodell eines parallelen Software-Systems als erstes Artefakt im Software-Entwicklungsprozess ansieht. Das Verfahren adressiert die Entwicklung nebenläufiger Software als architekturzentriertes modellgetriebenes System (MDS) und betont die Einrichtung von Software-Produktlinien und Familien.

Veröffentlicht als:
Stefan Gudenkauf. Domain-Specific Modelling for Coordination Engineering, Kiel Computer Science Series, 2012, (Books on Demand, ISBN 978-3-8482-2848-5). Online unter http://eprints.uni-kiel.de/20784/