

Definition und Anwendung von Logging-Systemen der zweiten Generation

Die Entwicklung komplexer Softwaresysteme wird immer aufwändiger. Die detaillierten Abläufe in Softwarelösungen und das Systemverhalten sind oft nur schwer nachvollziehbar. Eine professionelle Logging-Lösung der zweiten Generation kann Entwicklern und Projektmanagern umfassende Informationen aus dem „Inneren“ ihrer Software liefern.

Bei genauerer Betrachtung des Alltags eines Softwareentwicklers, der an einem komplexeren System mitarbeitet, ist die Aufzeichnung und Auswertung von Abläufen in der Software ein dauernd wiederkehrendes und brennendes Problem. Genau genommen handelt sich dabei um einen oft nicht bewusst wahrgenommenen, akzeptierten Engpass, durch den die Produktivität stark gehemmt wird.

Bis jetzt beschränken sich die meisten Softwareentwickler auf die Ausgabe der Daten in Logdateien oder in andere Speichermedien. Für die Auswertung erhält der Entwickler kaum Unterstützung. Oft erfolgt sie mit normalen Texteditoren, mit Kommandozeilen-Tools wie „grep“ oder mit für den speziellen Anwendungszweck geschaffenen Skripten.

Professionelle Logging-Lösungen versprechen hier eine nachhaltige Produktivitäts- und Qualitätsverbesserung, und zwar in allen Phasen, angefangen von der eigentlichen Entwicklung bis hin zum Betrieb der Softwaresysteme.

1 Anwendung einer Logging-Lösung innerhalb des Software-Lebenszyklusses

Typische Phasen des Software-Lebenszyklusses
<ul style="list-style-type: none">• Entwicklung und Einzeltest (Unittest)• Integration und Systemtest• Inbetriebnahme und Abnahme• Betrieb und Wartung

Bereits während der **Entwicklung** können durch die Visualisierung der Abläufe zur Laufzeit durch die Softwareentwickler schnellere Einzeltests durchgeführt werden. Software-Entwickler haben so einen größeren Überblick als im Debugger und können alle Vorgänge in Echtzeit verfolgen. Das aufgetretene Fehlverhalten wird im Detail aufgezeichnet und ist so besser analysierbar. Auch wenn ein Fehler erst später oder in anderen Zusammenhängen auftritt, werden die Informationen gesichert, die der Entwickler für die Analyse benötigt. Durch die Protokollierung der Kommunikation mit der Software (Programmen, Subsystemen, Komponenten) anderer Entwickler, Teams oder Hersteller wird auch die **Integration** inklusive Nachweisführung gegenüber Unterlieferanten erheblich erleichtert.

Während der **Inbetriebnahme** kann eine professionelle Logging-Lösung bei der Überwindung von Initialisierungs-, Konfigurations-, Netzwerk- und Geräteproblemen behilflich sein. Gerade die hier auftretenden Fehler sind oft besonders hartnäckig, wenn sie nur sporadisch auftreten. Auch Probleme aufgrund veränderter Rahmenbedingungen können schneller erkannt werden. Bei der **Abnahme** spielen die Aufzeichnungen der Abläufe bei der Erläuterung des oft sehr komplexen Systemverhaltens und dem Nachweis korrekter Funktion eine wichtige Rolle.

Nach erfolgreicher Inbetriebnahme einer neuen Software ermöglicht eine professionelle Logging-Lösung die **Inspektion** des aktuellen Systemzustands (auch für Telefon-Support, Fernwartung). Die **Aufzeichnung** von Abläufen im realen Einsatz kann für spätere Problemanalysen genutzt werden. In Projekten kann auch von *Kundenseite* aus gefordert sein, dass die Nutzereingaben, ggf. inklusive Verarbeitung und Ergebnissen, protokolliert werden.

2 Logging-Systeme zur Protokollierung und Auswertung von Abläufen

Die Protokollierung von Abläufen (kurz „Logging“) und deren Auswertung können je nach eingesetztem System unterschiedlich leistungsfähig sein

2.1 Logging-Systeme der ersten Generation

Logging-Systeme der ersten Generation beschränken sich auf die Ausgabe in Logdateien oder andere Speichermedien und unterstützen in keiner Weise eine weitreichende Analyse. In manchen Fällen mag das eine ausreichende Vorgehensweise sein, etwa für die Auswertung der Logdateien von FTP- oder Web-Servern, in denen die Kommunikation mit der Außenwelt protokolliert wird, welche über relativ einfache und feststehende Schnittstellen und Formate erfolgt.

Im Rahmen der Softwareentwicklung spielen die in den oft komplexen Systemen *intern* ablaufenden Prozesse eine viel größere Rolle, und es gibt meist viele beteiligte und sich ändernde Komponenten und Schnittstellen, so dass Arbeitsweisen wie die oben genannten hier sehr ineffizient sind.

Erwähnenswert ist jedoch, dass es für Logging-Systeme der ersten Generation neuerdings einige interessante Ansätze zur Standardisierung und Verbreitung gibt:

- Java 1.4 (Sun) enthält ein Logging-API und ebenso einige einfache „Loghandler“, d.h. Ausgabetreiber.
- Das .NET-Framework (Microsoft) enthält ebenfalls Logging-Unterstützung, aber in noch rudimentärerer Form.

Diese Beispiele zeigen, dass die Bedeutung von Logging-Funktionen zunehmend erkannt wird. Allerdings liegt der Fokus hier lediglich auf der Sicherung der Informationen.

2.2 Logging-Systeme der zweiten Generation

Logging-Systeme der zweiten Generation gestatten eine flexible Einflussnahme auf Art und Umfang der Protokollierung, ohne dass dafür der Quellcode der zu untersuchenden Anwendung jeweils geändert werden müsste, und ermöglichen Änderungen an den Einstellungen idealerweise gleich zur Laufzeit, ohne dass die Anwendung zum Aktivieren der Einstellungen neu gestartet werden müsste. Auch sollte sich der Nutzer nicht um Fragen wie Performance, Filterung, Entkopplung der Ausgaben, Ressourcenmanagement (z.B. Löschen alter Informationen) und Weiterleitung der Informationen kümmern müssen.

Noch wichtiger ist, dass zu einem Logging-System der zweiten Generation des weiteren ein interaktives Visualisierungs- und Analyse-Tool gehören muss. Mit ihm lässt sich die Auswertung für den Nutzer stark vereinfachen *und* um Größenordnungen beschleunigen.

Ein weiteres Muss ist die Bereitstellung von Filtermöglichkeiten für die Informationen. Unter anderem ist zu fordern:

- Filterausdrücke sollten visuell definiert werden können.
- Die Filter sollten logisch beliebig komplex sein können.
- Filter sollten auf anderen Filtern aufbauen können.
- Die Filter sollten abgespeichert werden können.
- Die Anwendung der Filter sollte komfortabel z.B. über Drag & Drop erfolgen.
- Die Anzeige der gefilterten Daten muss performant sein.

Im Rahmen einer aktuellen Fragestellung sollte der Benutzer bereits existierende Filter verwenden und darauf aufbauend schnell neue zusammenstellen können. Typischerweise beginnt er mit einer relativ groben Filterbedingung und blendet dann schrittweise weitere Informationen aus oder wieder ein, wobei er die Ergebnisse jeweils schnell überprüfen kann.

Die Informationen sollten nicht nur orts- und zeitversetzt (offline) angezeigt und ausgewertet werden können, sondern auch live (online). Durch Nutzung der o.g. Filtermöglichkeiten kann man dann so auch laufende Prozesse unter verschiedenen Aspekten *beobachten*.

Logging-Systeme der zweiten Generation beruhen insgesamt also auf der Erkenntnis, dass es nicht nur um das Sichern von Informationen geht, sondern *vor allem* auch um eine schnelle und flexible Anzeige und Auswertung. Das erinnert an die bekannte Idee der „Information at your fingertips“, wie Bill Gates es einmal anschaulich ausdrückte.

Theoretisch kann der Anwender ein Logging-System der ersten Generation zur zweiten Generation aufrüsten, indem er sich ein entsprechendes, kompatibles Analyse-Tool beschafft oder selber entwickelt. In der Praxis besteht die erste Option jedoch üblicherweise nicht, und die zweite ist meist nicht mit vertretbarem Aufwand realisierbar und attraktiv.

Steht die Entscheidung über den Einsatz eines Logging-Systems noch bevor, sollten folgende Fragen gestellt werden:

- Genügt ein System der ersten Generation den Anforderungen?
- Gibt es für dieses System Erweiterungen durch Analyse-Tools oder können solche demnächst erwartet werden?
- Erfüllt das System die technischen Anforderungen?
Zum Beispiel: einfache Schnittstelle, Unterstützung mehrerer Logkanäle, konfigurierbare Filterung der Ausgabe, Weiterleitung, automatisches Löschen alter Daten, Threadsicherheit, Performance

3 Auswahl und Einführung eines Logging-Systems

Ein im Rahmen eines beginnenden oder laufenden Projekts eingeführtes Logging-System muss sich in der Regel im Rahmen dieses Projekts rentieren, d.h. die Gesamtaufwände und insbesondere die Entwicklungszeit des Projekts verringern. Für eine schnelle Einführung ist neben einer einfachen Installation vor allem ein hohes Maß an intuitiver Bedienbarkeit gefordert. Dies ermöglicht nicht nur eine effizientere tägliche Arbeit mit dem Tool, sondern darüber hinaus eine schnelle und selbständige Einarbeitung der am Projekt beteiligten Entwickler ohne zusätzlichen Aufwand für externe Schulungen.

Eng verbunden mit der Frage der Verwendbarkeit sind auch Performance-Fragen. Erstens ist anzumerken, dass der Einsatz einer Logging-Komponente nicht zu nennenswerten Performance-Einbußen oder ungewollten Systembeeinflussungen oder gar zu Instabilitäten des gesamten Systems führen darf.

Zweitens sollte, wie angedeutet, auf die durch das Logging gesicherten Informationen schnell zugegriffen werden können, weshalb ein leistungsfähiges Tool für die Auswertung der schnell anwachsenden und oft unüberschaubaren Informationsmengen zum Umfang gehören sollte.

Bei der Einführung eines Logging-Systems sollte auf jeden Fall schrittweise vorgegangen werden. Soweit vorhanden, empfiehlt es sich, von den Herstellern bereit gestellte Evaluierungsversionen zu nutzen.

Ein erstes Versuchsfeld kann die Protokollierung der Vorgänge an den externen Schnittstellen des Anwendersystems sein, beispielsweise zu Nachweiszwecken. Eine andere Möglichkeit zur Erprobung der Praxistauglichkeit ist die Anwendung bei einer aktuell drängenden Fehlersuche.

iTech Logging 2

Hersteller: iTech Software GmbH, Berlin (<http://www.itech-software.de>)

Plattform: Windows NT oder höher (NT, 2000, XP), Windows 95 oder höher (95, 98, ME)

Logging-Bibliotheken:

- ITLogLib/.Net: Variante als .NET Assembly (DLL) für Windows .NET Sprachen, wie z.B. Visual Basic .NET und C#
- ITLogLib/COM: Variante als COM-DLL für alle Programmier- und Skriptsprachen, die COM-Objekte verwenden können (z.B. Visual Basic, VBA, Visual J++, VBScript und JScript)
- ITLogLib/Visual C++: Variante als DLL speziell für Microsoft Visual C++
- ITLogLib/Standard C++: Variante als DLL für Standard C++ Compiler (Borland, GNU, etc.)
- ITLogLib/Java: Variante für (pures) Java unter Windows
- ITLogLib/Delphi: Variante für Delphi
- ITLogLib/Win32: Standard Windows DLL (mit prozeduraler Schnittstelle)

Tools:

- Konfigurationstool ITConfigManager
- Analysetool ITLogBook

Lizenzierung:

- pro Entwickler