Wozu Programmier-Schablonen?

„Gehen Sie in das Gefängnis. Begeben Sie sich direkt dorthin. Gehen Sie nicht über Los. Ziehen Sie keine 4000 Euro ein." Weniger nostalgisch, dafür aber auf die wesentlichen Fakten reduziert, könnte Ihre „Ereigniskarte" auch lauten: „Spielstein auf Gefängnisfeld setzen."

Von Sonja Telscher, G.I.B

Unterschiedliche Vorlieben für Prosa oder Puristik gibt es auch in der Welt der Software-Entwickler. Während aufgeblähte Varianten das Programm verlangsamen, können bei zu kurzen Codings relevante Informationen fehlen und das Programm unbrauchbar machen. Blumige Quellcodes erschweren es anderen Entwicklern, sich im Programmablauf zurechtzufinden. Das sorgt für Probleme bei Neueinstellungen und Krankheitsausfällen.

Zusätzlich hat jeder Entwickler eigene Vorstellungen von einem ansprechenden und funktionellen Design: Der eine findet eine Werkzeugleiste am unteren Bildschirmrand praktisch, der andere versteckt sie lieber hinter einer Kachel oben rechts. Um dies zu vermeiden, gibt es Richtlinien und Beispiel-Codings. Sie sollen dafür sorgen, dass der jeweils optimale Code verwendet wird und eine einheitliche Struktur im Screen-Layout gesichert ist. Im Idealfall optimiert der effiziente Code die Rechnerleistung, erleichtert die Einarbeitung neuer Entwickler, ermöglicht eine problemlose Anpassung, Verschlankung oder Erweiterung der Anwendung und führt den User intuitiv durch das Programm.

Die Vorgaben können unternehmensspezifisch festgelegt werden (Eigenentwicklungen) oder man nutzt am Markt verfügbare Beispielprogrammierungen, die es in nahezu allen gängigen Sprachen wie Java, HTML oder C gibt. „Natürlich ist jeder gute Programmierer in der Lage, gewünschte Funktionen und Buttons zu programmieren. Aber dem Coding den letzten Schliff zu geben, also die effizienteste Syntax herauszuarbeiten, kostet Zeit. Wir haben deshalb bereits vor zehn Jahren angefangen, Programmier-Schablonen für den internen Gebrauch zu entwickeln, durch deren stringenten Einsatz wir über 40 Prozent Zeit einsparen konnten", so Felix Grab, Abap-Entwickler bei G.I.B. Auf einfache Art und Weise wurden somit die Softwareprodukte bei G.I.B mit optimiertem Coding und in einem einheitlichen Screendesign entwickelt. Das Interesse auf Kundenseite ließ nicht lange auf sich warten, sodass seit 2005 diese Programmier-Schablonen unter dem Namen „G.I.B Abap Programmier-Templates" am Markt erhältlich sind. „Der Erfolg unserer Abap-Templates erklärt sich auch dadurch, dass es für den Bereich Reporting und Auswertung keine Alternative gibt. Wer im Z-Namensraum Programme entwickelt und dabei ein einheitliches Look & Feel sowie eine stimmige und strukturierte Programmzusammensetzung sicherstellen möchte, muss entweder eigene Standards entwickeln oder er landet bei uns", führt Grab aus. Das Besondere an den Programmier-Schablonen der G.I.B ist dabei, dass sie jedes denkbare Szenario im Einsatzbereich abdecken. Der Programmierer kopiert nicht Ausschnitte eines Beispiels in sein Programm, er installiert die Templates komplett und modifiziert und individualisiert nur noch ausgewählte Bereiche.

Sonja Telscher ist Head of Marketing bei G.I.B.

Der Umfang der Programmier-Schablonen aus dem Hause G.I.B hat sich in den letzten zehn Jahren um das 15-Fache erhöht. „Wir sind mit einfachen Grids gestartet, die die alten Write-Listen abgelöst haben. Heute bieten unsere Schablonen Trees und Container, grafische Darstellungen, Auswertungen in beliebiger Anzahl auf mehreren Bildschirmen, Pop-up-Logiken, Drag-and-drop-Funktionen und Editiervorlagen", weiß Alexander Falge, Entwickler bei G.I.B.

Doch der Sprung von der Eigennutzung zur verkaufsfähigen Schablone ist groß. „Die Transformation in eine allgemeingültige Schablone lässt sich ungefähr mit dem dreifachen Aufwand beziffern", erklärt Falge. „Zunächst wird der fixe Rahmen im Coding festgestellt. Dafür sollten sämtliche Szenarien geprüft werden, in denen die Schablone eingesetzt werden könnte. Außerdem müssen Code und Dokumentation klar, einfach und unmissverständlich sein. Der Anwender soll die Templates sofort verstehen und einsetzen können. Und das in jeder Branche."

Die Entwickler stellen sich dieser Herausforderung mit dauerhaftem Engagement und haben selbst nach zehn Jahren Spaß an der Entwicklung. „Die Herausforderung, eine komplexe Logik für eine Vielzahl von Szenarien nutzbar zu machen, fasziniert mich", begeistert sich Falge. „Momentan stellen wir uns den neuen Herausforderungen Hana und UI5. Dabei konnten wir bereits sicherstellen, dass alle unsere Templates auf Hana problemlos laufen. Jetzt denken wir über spezifische Schablonen nach, die die Vorteile der In-memory-Logik voll ausnutzen, und natürlich über Vorlagen, mit denen unsere Kunden eigene Screens im Fiori-Design entwickeln können. Es bleibt also spannend bei uns!"

Bitte beachten Sie auch den Community-Info-Eintrag ab Seite 99