Info Bridge

Herausgeber: Object Engineering GmbH • www.objeng.ch • Juni 2014 • Nr. 10

Editorial

Liebe Leserinnen, liebe Leser

Es ist an der Zeit wieder einmal die Informations-Brücke zu Ihnen, liebe Leser zu schlagen. Die Zeit lief im Eilschritt und es ereigneten sich viele Dinge, welche die Schreibaktivität etwas untergehen liessen. Auch unsere Info Bridge wurde etwas modernisiert, sie bleibt aber nach wie vor auf vier Seiten im gleichen Format, der Druck wurde jedoch den heutigen Standards angepasst. Wir hoffen trotzdem, dass es nicht den Eindruck einer Hochglanzbroschüre hinterlässt, sondern das vermittelt, was der Zweck ist: Wissenswertes, technische Informationen und persönliche Erfahrung, die Ihnen hilfreich erscheinen.

In den letzten Jahren ist der technologische Fortschritt weiter vorangeschritten. Vor allem auf der Hardwareseite sind wieder viele Errungenschaften erzielt worden. Computer haben Solid State Disks, ohne mechanisch bewegte Teile, mehrere Prozessorkerne sind an der Tagesordnung und werden von der Software unterstützt, 3D Drucker werden langsam erschwinglich. All dies eröffnet neue Anwendungsmöglichkeiten.

Und wie steht es mit der Software Entwicklung?

Offen gesprochen bin ich da etwas skeptisch. Neuere Trends und Entwicklungsprozesse führen zu einem Adhoc-Vorgehen. Programmierung ohne Design (Architektur mit eingeschlossen) ist seit vielen Jahren im Trend. Einige Universitäten lehren die Studenten, dass UML quasi nur für's Studium und nicht für die Industrie sei, es werde "da draussen" nicht gebraucht. Die Fachhochschulen machen es im Hinblick auf praktische Umsetzung aus meiner Sicht besser. Dabei ist mir bewusst, dass dies möglicherweise etwas verallgemeinert ist. Ich sehe die Vorteile einer agilen Entwicklung sehr wohl und weiss, dass Sprints (Zielsetzungen über zwei Wochen) während der wirklichen Realisierung auch sinnvoll sind, um überhaupt je ans Ziel zu gelangen. Aus meiner Erfahrung aber bleibt Architektur und Design mehr oder weniger auf der Strecke. Speziell wenn man etwas Grosses und Komplexes aus vielen kleinen Teilen zusammensetzt, braucht es einen Konstruktionsplan und eine Gesamtübersicht. Können Sie sich vorstellen eine grosse Produktionsmaschine oder eine Fünf-Millionen Villa ohne Bauplan zu realisieren? Ein Architekt würde vermutlich zum Mond geschossen, wenn er dies so versuchen würde. Und mit welcher Begründung sollte es in der Software Entwicklung nun anders sein? Bei allen Methoden ist es jedoch nicht nur die eine Methode, sondern die richtige Mischung. Zudem ist eine gute Portion von Menschenverstand notwendig, um ans Ziel zu gelangen. Die heutigen Softwaresysteme sind aufgrund der Anforderungen, die sie erfüllen sollen schon komplex genug, durch mangelhafte Architektur und Designs sind sie meiner Meinung nach komplexer gebaut als nötig. "Zuerst denken, dann tun" ist nicht mehr in. Was muss passieren, damit man sich wieder darauf besinnt und weiter denkt, als nur bis zum Ende des laufenden Projektes?

Ich hoffe Sie finden etwas Zeit und Musse, sich die Info Bridge zu Gemüte zu führen, freuen sich am Erfolg unseres ehemaligen Mitarbeiters, lassen sich zum Rätseln beim Brückenwettbewerb hinreissen oder finden sonst eine nützliche Information.

Über Ihr Feedback freuen wir uns immer.

Herzlichen Gruss
Andres Koch

Ehemaliger Mitarbeiter schafft es zu Google

Interview mit Roman Zulauf

Du warst mehr als zehn Jahre ein treuer Mitarbeiter der Object Engineering. Fast vier Jahre ist es her, dass du die Schweiz verlassen hast, um in Los Angeles dein Master Studium in Computer Science zu absolvieren. Wie gefällt es dir mittlerweile in Los Angeles?

Zuerst musste ich einen ziemlichen Kulturschock überwinden. Mittlerweile habe ich mich an den amerikanischen Lifestyle gewöhnt und es gefällt mir sehr gut. Kalifornien ist ein wunderschöner Staat und das Klima ist traumhaft. Die Stadt Los Angeles ist kulturell sehr durchmischt und hat für jeden etwas zu bieten.

Unterdessen hast du dein Masterstudium erfolgreich abgeschlossen und arbeitest nun für Google. Damit hast du einen riesigen Karrieresprung gemacht. Zuerst einmal herzliche Gratulation! Wie kam es dazu?

Eigentlich war ich nicht auf Jobsuche. Ein Studienkollege aus der Algorithmen-Vorlesung durchlief gerade bei Google den Rekrutierungsprozess. Während unserem täglichen Jogging zwischen Vorlesungen besprachen und lösten wir verschiedene sehr interessante theoretische Berechnungsprobleme in bestmöglicher Laufzeit. Dies schürte mein Interesse in theoretische Informatik und Komplexitätstheorie. Einige Monate später fragte mich mein Kollege, inzwischen ein Google-Mitarbeiter, nach meinem Lebenslauf. Mittlerweile war ich überzeugt, dass Google für mich eine logische Fortsetzung meiner Karriere darstellen würde. Einige technische Interviews später erhielt ich dann ein Angebot für die Position Software Engineer.

Fortsetzung Seite 4

Risiko und Chancen bei der Software Modernisierung

Im letzten Beitrag (Info Bridge Nr. 9) haben wir die Ablösung von Legacy-Applikationen und bewährte Vorgehen betrachtet. In diesem Artikel möchten wir etwas für die Modernisierung von Software sensibilisieren bevor das Unterfangen zum Himmelfahrtskommando wird. Zu oft hören wir "ja aber es läuft doch alles bestens, wir hatten seit längerem kein Probleme mit dieser Software". Dies hört sich nach der Aussage einer Person an, welche quasi auf einem offenen (aber gefüllten) Pulverfass sitzt und gemütlich ihre Pfeife stopft, anzündet und zu rauchen beginnt, während die letzten fachkundigen Mitarbeiter gerade einen neuen Job bei einem anderen Unternehmen annehmen oder in den "wohlverdienten" Ruhestand treten. Sind sich die Verantwortlichen des Risikos des „Pulverfasses" bewusst oder muss der Betrieb erst einige Tage stillstehen? Wir verkaufen keine Versicherungen, also lassen wir einmal die Schwarz-(pulver)malerei beiseite und zeigen auf, was man tun kann.

Bewährtes Vorgehen

Beabsichtigt man ein bestehendes System oder eine Software zu modernisieren, ist der erste Schritt immer die Aufnahme der Ist-Situation, also eine gründliche Analyse. Erst dann erstellt man den Vorgehensplan und wählt individuell die geeigneten Verfahren aus.

Analyse, aber wie tief und detailliert?

Nimmt man ein solches Software-System unter die Lupe, stellt man fest, dass dieses meistens seit Jahren stabil läuft und einen wichtigen, individuellen Teil des Geschäftsprozesses abdeckt. Da heisst es zuerst Daten sammeln. Dabei deckt man die unscheinbaren oder unentdeckten Risiken auf.

Wichtig ist, sich das Domain-Wissen anzueignen, um überhaupt einschätzen zu können, wie wichtig diese Software für das Unternehmen ist. Danach sollte man eine Person finden, welche dieses Programm und dessen Funktionen bestens kennt, sie befragen, bis man möglichst viel Wissen über das System hat. Dokumentationen können helfen, aber auch irreleiten. Verwundern Sie sich aber nicht, wenn diese seit Jahren nicht mehr nachgeführt wurden, obwohl Änderungen am Programm vorgenommen wurden.

Gehen wir einmal davon aus, dass wir uns selber in die Programme einlesen können, und die Dokumentationen und verwendeten Normen einigermassen akkurat sind, dann findet das geübte Auge bald das eine oder andere Muster, mit welchem man nach einer 80:20 Regel den meisten Code analysieren kann. Das Ziel müsste nämlich sein, aus dem Programmcode Abhängigkeiten zu den einzelnen Modulen, Mengengerüste, Strukturen (z.B. Datenbank- oder Datei-Records) und zusätzliche Informationen systematisch zu gewinnen, welche später als Metadaten (beschreibende Eigenschaften von Daten) für die Modernisierung verwendet werden können. Dies ist insbesondere dann wichtig, wenn es sich nicht nur um Programme handelt, welche gesamthaft nicht nur einige Tausend Code-Zeilen beinhalten, sondern wenn es um Grössenordnungen von Zehn-Tausend, Hundert-Tausend oder sogar Millionen geht. Ohne eine weitgehende Automatisierung ist man zu ungenau und zu langsam. Daten, wie Konfigurations-Dateien und -Datenbanken, darf man dabei nicht vergessen. Nicht selten werden solche Kernprogramme bereits durch Metadaten gesteuert, was aus den eigentlichen Programmen nur schwer herauszulesen ist.

Erst wenn man eine Fülle von relevanten Informationen über das betrachtete System hat, kann man das Risiko einer Ablösung resp. Nicht-Ablösung für das Unternehmen abschätzen. An dieser Stelle hat man in der Regel bereits eine gute Vorstellung was nötigenfalls (wenn überhaupt) gemacht und was nicht gemacht werden soll.

Ohne Migrations-Architektur keinen Schritt weiter

Mit den aus der Analyse erhaltenen Informationen und mit der Zielsetzung, das Ausfallsrisiko zu vermindern, kann man nach organisatorischen und technischen Lösungen suchen. In einer ersten Phase bleibt man eher informal, sollte jedoch strukturiert vorgehen, wenn es gilt, technische Massnahmen zu treffen.

Man geht dazu über, eine grobe System- und Software-Architektur zu erstellen, was bei der Wahl des Vorgehens massiv hilft, sowie auch dabei unterstützt, das Projekt zu planen, schrittweise die Umsetzung durchzuführen ohne den Betrieb zu stören. Ohne eine klare Architektur ist es etwa so, wie wenn man eine Stadtumfahrung plant, ohne eine Umleitung des bestehenden Verkehrs einzuplanen. Die Architektur gibt die Leitlinie, die Struktur des neuen oder erneuerten Systems definiert die verwendeten Methoden, Technologien und Werkzeuge. Daraus folgen dann die Realiserungsschritte und die Releasepläne.

Das richtige Verfahren wählen

Nach der Erstellung der Migrations-Architektur sieht man meist eine oder mehrere Möglichkeiten, wie man das System auf eine modernisierte Umgebung umstellen kann. Es gilt da natürlich die Kosten abzuschätzen und das Verfahren zu wählen, welches mit geringstem menschlichem Zutun die besten Resultate liefert. Da ist Automatisierung angesagt, wenn überhaupt möglich.

Wrapping, wenn es um Integration geht

Wrapping heisst eigentlich "Einpacken". Bei der Modernisierung ist es ein Verfahren, wie man zu einem bestehenden (alten) System eine Fassade erstellt, über die andere Systeme Daten austauschen können, also integriert werden können. Dabei achtet man darauf, dass die funktionalen Anforderungen bestimmen, welche Schnittstellen nötig sind. Auf mehr sollte man verzichten. Hinter dieser Fassade kann eine echte Ablösung durchgeführt werden.

Refactoring, wenn die technische Grundlage nicht zu alt ist

Oft müssen Programme, welche vor 8-15 Jahren erstellt wurden, massiv überarbeitet werden. In solchen Zeitabschnitten ändern entweder die Software-Technologien oder die darunterliegenden Plattformen. Manchmal sind es ausgelassene Versions-Aktualisierungen, welche nicht berücksichtigt werden wollten oder konnten. Dann müssen die Softwarekomponenten auf neue Versionen angehoben werden oder Bibliotheken müssen auf den neusten Stand gebracht werden, was oft auch Änderungen des API (Application Programming Interface) mit sich bringt. Man nennt diese Arbeit *Refactoring* oder auch Restrukturierung oder Umgestaltung. Neue Technologien oder Bibliotheken können selbstgeschriebenen Code ablösen und die Wartbarkeit erhöhen. Insbesondere können Open-Source-Komponenten, die typischerweise defacto-Standards sind oder sich an bestehenden Standards orientieren, eigene Komponenten ersetzen.

Plattform-Portierung, wenn die bestehende zu kostspielig ist

Besonders wenn alte Systeme im Spiel sind, ist mit hohen Wartungs- und Lizenz-Kosten zu rechnen. Das Risiko, dass das System von heute auf morgen stillsteht und ein Hardwareteil ausfällt, welches auf dem Markt gar nicht mehr erhältlich ist, kann einem einen Schauer über den Rücken jagen. Fällt zum Beispiel eine Harddisk aus, die in der gewünschten kleinen Kapazität nicht mehr verfügbar ist, könnte es sein, dass die Betriebssystem-Software Ersatz-Disks mit viel grösserer Speicherkapazität, gar nicht mehr behandeln kann. Dann ist guter Rat teuer. Deshalb sollten alte Computer, welche ähnliche Komponenten enthalten, wie andere im Einsatz stehende Computer, nicht entsorgt werden, da sie später einmal die Lösung für ein unerwartetes Problem sein könnten.

Meistens ist nicht nur die Applikationssoftware betroffen, sondern auch die Werkzeuge wie Compiler, Datenbanksysteme, Betriebssystem und Hilfsprogramme, welche auf einer neuen Hardware-Plattform nicht mehr verfügbar sind. Software, welche für 32-Bit-Betriebssysteme geschrieben wurden, funktionieren oft auf einer 64-Bit-Architektur nicht mehr.

Emulationen oder virtuelle Systeme stellen in gewissen Fällen eine gute Lösung dar. Alte Maschinen werden auf einer neuen Plattform durch ein spezielles Programm emuliert und dank der x-fach höheren Hardwaregeschwindigkeit läuft dieses Programm sogar noch schneller, als auf der Original-Hardware. Aber auch solche Emulationen werden irgendeinmal "alt" und müssen abgelöst werden. Eine solche Lösung dient meist nur als Übergangslösung für eine nicht allzulange Zeit.

Sprachportierung

Wenn viel guter Code vorhanden ist oder Metadaten für die Generierung verwendet werden können, kann auch eine Sprachportierung in Betracht gezogen werden. Neue Programmiersprachen sind verlockend, da bekannt, gelehrt und durch Werkzeuge gut unterstützt. Oft sind Programme, die in Sprachen von früheren Generationen geschrieben wurden, in grosser Quantität und relativ guter Qualität vorhanden. Der Portierungs-Aufwand darf jedoch nicht unterschätzt werden. Dabei ist zu beachten, dass solche Übersetzer nicht den ganzen Sprachumfang abdecken müssen, da von der Programmierung oft nur ein beschränkter Teil verwendet wurde. Dagegen kann nicht erwartet werden, die gleiche Untermenge, die in einer Firma verwendet wurde, bei einer anderen wieder anzutreffen. Ältere Sprachen sind oft auch eng an Betriebssysteme gekoppelt und nutzen deren System-Bibliotheken sehr gut aus. Neuere Sprachen sind eher loser angebunden. In solchen Fällen kann man aber immer Metadaten aus den bestehenden Ressourcen gewinnen.

Aus einem in einer algorithmischen Sprache geschriebenen Programm wird auch durch eine Übersetzung in eine moderne Sprache kein objektorientiertes Programm entstehen. Dazu gehört mehr. In vielen Fällen reicht es aber aus, daraus ein objektbasiertes Programm zu machen, oder ein Programm, das die Objektorientierung nutzt, aber nur einfache Teile (z.B. Unter-Klassen) für ein neues Framework generiert.

Dank moderner Parser-Technik können bestehende Programme zuerst in einen abstrakten Syntax-Baum (AST) umgesetzt werden, woraus dann verschiedene Artefakte generiert werden können. Flexible Parser und Generatoren sind unentbehrliche Werkzeuge, ebenso das Know-how, diese effizient einzusetzen. Je automatischer eine Umsetzung gemacht werden kann, um so höher wird die Qualität ausfallen.

Fazit

Bei einer Entscheidung über das Vorgehen, ist auch die Dauer, wie lange die modernisierte Software weiter im Betrieb bleiben soll, ausschlaggebend. Soll das Programm nur für weitere 5 Jahre im Einsatz bleiben, dann lässt sich nicht jeder Aufwand vertreten. Aber Achtung: Totgesagte leben oft länger und neue Projekte können dauern, werden zu spät initialisiert oder fallen einer Sparrunde zum Opfer. Dann sollten die "alten" Komponenten weiterhin die Aufgabe übernehmen. Mit einer Investitionsrechnung kann in solchen Fällen die Richtigkeit einer Modernisierung abgeschätzt werden. Ist der Ablösungszeitpunkt nicht mindestens auf ein Jahr genau voraussehbar, werden Modernisierungsvorhaben oft nicht gestartet, was sich später als Bumerang erweisen kann.

Eine Sache, die man bei allen verfügbaren Alternativen **nicht tun sollte, ist einfach zuzuschauen.** Aktivitäten, welche die relevanten und geschäftskritischen Systeme betreffen (von Analyse bis zu Modernisierung) sollten auf der mittel- oder langfristigen Planung erscheinen. Erst zu reagieren, wenn ein akutes Problem auftritt, kann eine Lösung in der verbleibenden Zeit verunmöglichen. Wenn man sich aber mit der Situation vertraut macht und die Risiken nur halbwegs richtig einschätzt, hat man die Chance mit weniger Aufwand als man meint, aber mit guten Ideen ein Desaster abzuwenden.

Aktuelles

Brückenwettbewerb

Die Wettbewerbsfragen lauten:
Wie heisst die abgebildete Brücke?
Was ist falsch in diesem Bild?

Ihre Antwort senden Sie bitte bis zum 31. Juli an:
infobridge@objeng.ch
Unter den Teilnehmern mit den richtigen Antworten wird eine passende Überraschung ausgelost.

Auflösung Wettbewerb Info Bridge 9

Der Name der Brücke lautet **Siebzehn-Bogen-Brücke.** Sie spannt sich mit ihren 150m Länge über den Kuming See und befindet sich im riesigen Areal des Sommerpalasts im Nordwesten von Peking. Die Siebzehn-Bogen-Brücke hat, wie ihr Name schon sagt, siebzehn Bogen und ist mit 564 Löwenfiguren aus Marmor dekoriert. Sie wurde 1750 von Kaiser Qian Long errichtet.

Heute ist das Areal des Sommerpalastes eine der schönsten Parkanlagen von Beijing (Peking). Im Jahr 1998 wurde der Sommerpalast von der UNESCO in die "Liste des Weltkultur- und -naturerbes" aufgenommen.

Der Gewinner des Brücken-Wettbewerbs der Info Bridge 9 heisst Jürg Elsasser. Er gewinnt eine chinesische Teetasse mit Kostproben von chinesischem Tee. Wir danken für die rege Beteiligung.

Fortsetzung Interview

Was genau arbeitest du bei Google und wie ist es da zu arbeiten?

Ich arbeite in der "Ads Quality" Abteilung und verwende statistische Methoden, um die Qualität, sprich Relevanz und User Experience der Werbung von Google zu optimieren. Es ist ein richtiges Erlebnis in einem der Top Tech-Unternehmen zu arbeiten. Die Unternehmenskultur ist hervorragend, mit viel Nachdruck auf Innovation. Dies macht die Arbeit sehr spannend, denn Google versucht immer an die Grenzen des Möglichen zu gehen. Es läuft immer enorm viel, und Dinge gehen mit einem Riesentempo vorwärts.

Zurück zur Object Engineering. Du hast da etwas mehr als zehn Jahre gearbeitet. Erzähle uns etwas von deinem Werdegang und den verschiedenen Projekten.

Als selbstgelernter Programmierer und Quereinsteiger bekam ich damals die einmalige Gelegenheit, bei der Object Engineering mich meiner Leidenschaft professionell zu widmen. Zuerst arbeitete ich an diversen internen Projekten, bald durfte ich an produktive Projekte ran. Gleichzeitig absolvierte ich berufsbegleitend eine formelle Ausbildung an der Fachhochschule. Die Kombination von Praxis und Theorie stellte ein ideales Szenario dar: Ich konnte die neu gelernten Fertigkeiten direkt umsetzen und die gewonnene Praxis erleichterte mir handkehrum das Studium. Die strikten technischen Normen innerhalb der Object Engineering und der Nachdruck auf eine systematische Arbeitsweise lehrten mich die notwendige Disziplin, die in einem professionellen Umfeld notwendig ist. Über die Jahre durfte ich in einigen interessanten Projekten mitwirken. Von User-Interfaces zu Daten-getriebenen Applikationen war von allem etwas dabei. Die verschiedenen Migrations- und Integrationsprojekte waren eine fast unerschöpfliche Quelle von neuen Herausforderungen.

Wie hast du deine Arbeit in der Object Engineering persönlich erlebt?

Die Arbeit bei der Object Engineering hat mir über die Jahre immer grossen Spass gemacht. Durch die vielen verschiedenen Projekte hatte ich die notwendige Abwechslung und konnte mich persönlich dauernd weiterentwickeln. Das Arbeitsumfeld und die Atmosphäre in der Object Engineering empfand ich als nahezu ideal.

Wie beurteilst du nun, mit etwas Abstand und neuen Erfahrungen in einer Riesenfirma wie Google, ein Unternehmen wie die Object Engineering und ihre Tätigkeiten?

Für den motivierten und kreativen Arbeitnehmer sehe ich ein solches Unternehmen als absolute Chance, denn hier kann man sich voll entfalten und direkten Einfluss auf vieles nehmen, was in grösseren Unternehmen durch die fixeren Strukturen und Prozesse schwieriger wird. Aus Kundensicht denke ich, dass ein Unternehmen dieser Art viel mehr Nähe zum Kunden zeigen kann. Und die Object Engineering hat bewiesen, dass hervorragende Qualität auch im kleinen Team erreicht werden kann.

Vielen Dank für das interessante Interview. Wir wünschen dir weiterhin viel Erfolg und Freude in deiner weiteren Laufbahn.

Good Luck Roman!

Object Engineering GmbH
Birmensdorferstr. 32
CH-8142 Uitikon-Waldegg

Tel: +41 (0) 44 400 47 00
Fax: +41 (0) 44 400 47 07

www.objeng.ch
info@objeng.ch