10 Leuchtspurmunition

Bereit, Feuer, Zielen ...

Es gibt zwei Wege, im Dunkeln mit einem Maschinengewehr zu schießen.[5] Man kann herausfinden, wo genau sich das Ziel befindet (Entfernung, Höhe und Richtung). Man kann dazu auch die Umgebungsbedingungen messen (Temperatur, Feuchtigkeit, Luftdruck, Windgeschwindigkeit und so weiter). Man kann außerdem die genauen Spezifikationen der verwendeten Patronen und ihre Wechselwirkung mit dem Gewehr nachschlagen. Man kann schließlich auch noch Tabellen oder einen Ballistikcomputer verwenden, um die exakte Zielrichtung und Höhe des Laufes auszurechnen. Wenn alles so funktioniert wie angegeben, die Tabellen korrekt sind und sich die Umgebungsbedingungen nicht ändern, sollte die Kugel ziemlich nahe am Ziel einschlagen.

Oder man verwendet Leuchtspurmunition.

Leuchtspurmunition wird in regelmäßigen Abständen zwischen die normale Munition in den Patronengurt geladen. Wenn sie abgeschossen wird, entzündet sich der Phosphor darin und hinterlässt eine leuchtende Spur vom Gewehr zu allem, was getroffen wurde. Wenn die Leuchtspurmunition das Ziel trifft, tut es die normale Munition auch.

Es überrascht nicht, dass Leuchtspurmunition mühsamen Berechnungen vorgezogen wird. Die Rückmeldung ist unmittelbar und externe Einflüsse werden minimiert, weil sie denselben Umgebungsbedingungen unterliegt wie die echte Munition.

Diese Analogie ist vielleicht brutal, aber sie gilt auch für neue Projekte – insbesondere, wenn Sie etwas bauen, das vorher noch niemand gebaut hat. Wie die Schützen versuchen Sie, ein Ziel zu treffen, das im Dunkeln liegt. Die Anforderungen der Benutzer sind vage, weil sie noch nie ein solches System gesehen haben. Sie stehen einer Menge Unsicherheiten gegenüber, weil Sie unbekannte Algorithmen, Techniken und Programmiersprachen verwenden. Projekte brauchen Zeit, Sie können sicher sein, dass sich Ihre Arbeitsumgebung verändern wird, noch bevor Sie fertig sind.

Die klassische Antwort darauf ist, das System zu Tode zu spezifizieren. Produzieren Sie meterweise Papier, das jede einzelne Anforderung auflistet, jede Unbekannte festlegt und die Umgebung einschränkt. Sie feuern die Waffe blind ab. Eine große Berechnung am Anfang, dann schießen und hoffen.

Pragmatische Programmierer bevorzugen jedenfalls Leuchtspurmunition.

[5] Genau genommen gibt es viele Wege, mit einem Maschinengewehr zu schießen, einschließlich mit geschlossenen Augen wild in die Gegend zu feuern. Aber es geht nur um die Analogie und da dürfen wir uns Freiheiten nehmen.

Quelltext, der im Dunkeln leuchtet

Leuchtspurmunition funktioniert, weil sie denselben Bedingungen unterliegt wie die normale Munition. Sie kommt schnell ans Ziel, so dass der Schütze unmittelbare Rückmeldung bekommt, und sie ist eine relativ günstige Lösung.

Um mit Quelltext den gleichen Effekt zu erzielen, brauchen wir etwas, das uns schnell, sichtbar und wiederholbar von einer Anforderung zu einem Aspekt des endgültigen Systems bringt.

Tipp 15: ***Verwenden Sie Leuchtspurmunition, um das Ziel zu finden*** ◄

Wir haben einmal ein komplexes Client-Server-Projekt für Datenbank-Marketing durchgeführt. Teil der Anforderungen war die Möglichkeit, Abfragen über Zeiträume zu definieren und auszuführen. Der Server bestand aus einer Reihe von relationalen und anderen spezialisierten Datenbanken. Der Client, in Object Pascal geschrieben, verwendete C-Bibliotheken für die Schnittstelle zu den Servern. Die Anfragen der Anwender wurden auf dem Server in einer Lisp-artigen Notation gespeichert, bevor sie unmittelbar vor der Ausführung in optimierte SQL-Anweisungen konvertiert wurden. Es gab viele Unbekannte und viele verschiedene Umgebungen. Niemand wusste genau, wie sich die Benutzeroberfläche verhalten sollte.

Das war eine gute Gelegenheit für Leuchtspur-Quelltext. Wir entwickelten das Framework für die Benutzeroberfläche, Bibliotheken für die Repräsentation der Anfragen und eine Struktur für die Umwandlung einer gespeicherten Anfrage in das datenbankspezifische Format. Dann haben wir alles zusammengebaut und ausprobiert, ob es funktioniert. In der ersten Version konnten wir nur eine Anfrage abschicken, die alle Zeilen einer Datenbanktabelle zurücklieferte, aber wir haben damit bewiesen, dass sich die Benutzeroberfläche mit den Bibliotheken versteht, die Bibliotheken Anfragen serialisieren und deserialisieren können und schließlich die Server aus dem Ergebnis SQL erzeugen können. In den folgenden Monaten füllten wir diese Grundstruktur schrittweise aus, indem wir gleichzeitig zu jeder Komponente des Leuchtspur-Quelltextes neue Funktionalität hinzugefügt haben. Wenn die Benutzeroberfläche einen neuen Anfragetyp bekam, wuchs die Bibliothek, und die SQL-Generierung wurde ausgeklügelter.

Leuchtspur-Quelltext ist kein Wegwerfartikel: Sie schreiben ihn mit ernsten Absichten. Er enthält alle Fehlerprüfungen, Strukturierungen, Dokumentation und Selbsttests, wie jedes andere Stück produktiv eingesetzter Quelltext auch. Er ist nur nicht voll funktionstüchtig. Aber sie haben eine Ende-zu-Ende-Verbindung aller Komponenten in Ihrem System erreicht und können sehen, wie nah Sie Ihrem Ziel gekom-

men sind, und gegebenenfalls nachsteuern. Wenn Sie das Ziel erst einmal erfasst haben, können Sie einfach Funktionalität hinzufügen.

Leuchtspur-Software-Entwicklung folgt der Idee, dass ein Projekt nie fertig ist: Änderungen und Erweiterungen sind immer nötig. Es ist ein *inkrementelles* Vorgehen.

Die konventionelle Alternative dazu ist ein schwerfälliger Konstruktionsprozess: Quelltext wird in Module unterteilt und in einem luftleeren Raum geschrieben. Module werden zu Baugruppen kombiniert, die ihrerseits wieder kombiniert werden, bis man eines Tages ein vollständiges Programm hat. Erst dann kann die Anwendung als Ganzes den Anwendern gezeigt und getestet werden.

Software-Entwicklung mit Leuchtspurmunition hat viele Vorteile:

- **Anwender bekommen früh etwas Funktionierendes zu sehen.** Wenn Sie das, was Sie tun, erfolgreich kommuniziert haben (siehe *Hohe Erwartungen*, Seite 240), wissen Ihre Anwender, dass sie etwas Unausgereiftes sehen und werden nicht wegen fehlender Funktionalität enttäuscht sein. Ganz im Gegenteil, sie werden begeistert sein, wenn sie erkennbare Fortschritte ihres Systems sehen, und sich im Laufe des Projektes immer mehr beteiligen und so ihr Engagement steigern. Diese Anwender werden diejenigen sein, die Ihnen nach jeder Iteration sagen, wie nah Sie dem Ziel schon sind.
- **Software-Entwickler erstellen eine Struktur, die sie ausfüllen können.** Das erschreckendste Stück Papier ist das, auf dem nichts steht. Wenn Sie alle Ende-zu-Ende-Interaktionen in Ihrem System ausgearbeitet und in Quelltext gegossen haben, muss sich Ihr Team nicht mehr so viel aus den Fingern saugen. Das macht alle Beteiligten produktiver und fördert die Konsistenz.
- **Sie haben eine Integrationsplattform.** Wenn das System von Ende zu Ende verbunden ist, haben Sie eine Umgebung, in der Sie neue Funktionalität ergänzen können, sobald diese die Modultests bestanden hat. Statt einer Big-Bang-Integration werden Sie täglich integrieren (oft sogar mehrmals täglich). Die Auswirkungen jeder neuen Veränderung sind damit viel offensichtlicher und die Wechselwirkungen begrenzter, so dass Fehlersuche und Tests schneller und genauer durchgeführt werden können.
- **Sie können etwas vorzeigen.** Die Auftraggeber des Projekts und die hohen Tiere neigen dazu, zu den unmöglichsten Zeiten Demonstrationen sehen zu wollen. Mit Leuchtspur-Quelltext haben Sie jederzeit etwas zum Vorzeigen.
- **Sie entwickeln ein besseres Gefühl für den Projekt-Fortschritt.** Bei Leuchtspur-Software-Entwicklung nehmen die Programmierer einen Anwendungsfall nach dem anderen in Angriff. Wenn einer fertig ist, gehen sie zum nächsten über. Damit wird es viel einfacher, den Fortschritt zu messen und für die Anwender sichtbar zu machen. Sie vermeiden damit diese riesigen monolithischen Quelltextblöcke, die Woche für Woche angeblich zu 95% fertig sind, weil jede einzelne Entwicklung klein und überschaubar ist.

Leuchtspurmunition trifft nicht immer ins Ziel

Leuchtspurmunition zeigt, was getroffen wird. Das muss nicht immer das Ziel sein. Sie passen dann Ihre Peilung an, bis Sie es im Visier haben. Darum geht es.

Mit Leuchtspur-Quelltext ist es genauso. Sie verwenden diese Technik in Situationen, in denen Sie nicht hundertprozentig wissen, wohin die Reise geht. Es sollte Sie nicht überraschen, wenn die ersten paar Versuche das Ziel verfehlen. Die Anwender sagen „Das ist nicht das, was wir gewollt haben.“, Daten, die Sie brauchen, sind nicht verfügbar oder es werden Performance-Probleme erkennbar. Überlegen Sie sich, wie Sie das, was Sie haben, verändern können, um näher an das Ziel zu kommen, und seien Sie froh, dass Sie dafür eine schlanke Entwicklungsmethode benutzt haben. Eine kleine Menge Quelltext hat wenig Trägheit und kann schnell und einfach geändert werden. Sie können mehr Rückmeldung bekommen und schneller als mit jeder anderen Entwicklungsmethode eine neue, genauere Version erstellen. Ihre Anwender können sich sicher sein, dass das, was sie sehen, der Realität entspricht und nicht nur auf Papier spezifiziert ist, weil jeder wichtige Bestandteil der Anwendung im Leuchtspur-Quelltext vertreten ist.

Leuchtspur-Quelltext oder Prototypen

Vielleicht denken Sie, das Konzept von Leuchtspur-Quelltext ist nichts anderes als ein plakativer Name für Prototyping. Es gibt einen Unterschied. Mit einem Prototypen, streben Sie an, bestimmte Aspekte des endgültigen Systems zu erkunden. Bei einem echten Prototypen werden Sie alles, was Sie bei Ihren Versuchen zusammengestrickt haben, wegwerfen und es mit den gewonnenen Erfahrungen noch mal programmieren.

Angenommen, Sie erstellen eine Anwendung, die Spediteuren hilft, Kisten mit ungewöhnlichen Maßen in Container zu stapeln. Neben anderen Problemen müssen Sie die Benutzerschnittstelle intuitiv gestalten und komplexe Algorithmen für das optimale Packen verwenden.

Sie könnten für die Anwender einen Prototypen der Benutzerschnittstelle mit einem Werkzeug bauen und gerade so viel ausprogrammieren, dass er auf Eingaben reagiert. Sobald die Anwender mit der Aufmachung zufrieden sind, würden Sie den Prototypen wegwerfen und ihn mit echter Funktionalität und in der gewählten Programmiersprache neu implementieren. Ebenso wollen Sie vermutlich verschiedene Algorithmen für das Packen ausprobieren. Sie könnten funktionale Tests in einer toleranten Hochsprache wie zum Beispiel Perl und Performance-Tests in einer niederen, maschinennahen Programmiersprache implementieren. Auf jeden Fall würden Sie, sobald Sie die Entscheidung getroffen haben, von vorne anfangen und die Algorithmen in der endgültigen Programmiersprache mit Schnittstellen zur realen Umgebung implementieren. Das ist *Prototyping*, und es ist sehr nützlich.

Mit Leuchtspur-Quelltext bearbeitet man ein anderes Problem. Man will wissen, wie die Anwendung als Ganzes zusammenhängt, den Anwendern zeigen, wie die Interaktionen in der Praxis funktionieren und den Software-Entwicklern ein Architekturgerüst geben, in das sie ihren Quelltext einbauen können. In diesem Fall würden Sie Leuchtspur-Quelltext erstellen, der aus einer trivialen Implementierung des Packungsalgorithmus (vielleicht so etwas wie „wer zuerst kommt, mahlt zuerst“) und einer einfachen, aber funktionierenden Benutzerschnittstelle besteht. Wenn alle Teile der Anwendung zusammengeschraubt sind, haben Sie einen Rahmen, den Sie Ihren Anwendern und Software-Entwicklern zeigen können. Mit der Zeit fügen Sie in diesen Rahmen neue Funktionalität ein, indem Sie Unterprogrammrümpfe ausfüllen. Der Rahmen bleibt dabei intakt und Sie können sich sicher sein, dass sich das System weiterhin so verhält, wie die erste Version Ihres Leuchtspur-Quelltextes.

Diese Unterscheidung ist wichtig genug, um eine Wiederholung zu rechtfertigen. Prototyping erzeugt Wegwerf-Quelltext. Leuchtspur-Quelltext ist schlank, aber vollständig und bildet einen Teil des Gerüsts für das endgültige System. Betrachten Sie Prototyping als Aufklärung bevor die erste Kugel Leuchtspurmunition verschossen wird.

Verwandte Abschnitte

- *Gut ist gut genug*, Seite 9
- *Prototypen und Post-it-Zettel*, Seite 48
- *Die Spezifikationsspirale*, Seite 204
- *Hohe Erwartungen*, Seite 240

11 Prototypen und Post-it-Zettel

In vielen Branchen werden Prototypen eingesetzt, um bestimmte Ideen auszuprobieren, denn Prototyping ist deutlich billiger als die Produktion in vollem Umfang. Automobilhersteller bauen beispielsweise viele verschiedene Prototypen eines neuen Autos, von denen jeder für die Erprobung eines bestimmten Aspekts des fertigen Autos gedacht ist – Aerodynamik, Aussehen, Konstruktionseigenschaften und so weiter. Für Tests im Windkanal wird vielleicht ein Tonmodell gebaut, für die Design-Abteilung reicht eventuell ein Modell aus Balsaholz und Klebeband aus und so weiter. Einige Automobilhersteller gehen noch einen Schritt weiter und arbeiten zu einem großen Teil mit Computermodellen, um die Kosten noch stärker zu senken. Auf diese Weise können riskante und unsichere Dinge ausprobiert werden, ohne dass das echte Objekt gebaut werden muss.