Projektmanagement

Projektmanagement aus Software-Sicht

Wie entwickeln, wie effizient nach Fehler suchen, wie Code strukturieren

Verschiedene Entwicklungsmodelle in Verwendung

Fokus heute: Projektmanagement aus Projektsicht, Erfahrungen aus Spieleentwicklung und Projektmanagement durch nicht-Entwickler

Teilprojektplanung

Projekt in Teile auspalten, für jeden Teil überlegen

- Wie viel Aufwand vermutlich notwendig?
- Wie wahrscheinlich, dass es mehr Aufwand wird, oder so viel, dass der Teil scheitert?
- Welche anderen Teilprojekte notwendig, damit dieser Teil entwickelt werden kann?
- Welche anderen Teilprojekte notwendig, damit dieser Teil einen Nutzen hat?

Woher Zahlen nehmen, wenn unbekannt: Frei Schätzen. Gibt man sich Mühe, passen Ergebnisse oft genau genug, Fehler heben sich teilweise gegenseitig auf. Vor allem: Bessere Werte bekommt man nicht.

Projektübersicht

Sind zu viele Teilprojekte vorhanden: Welche kann man im Grundplan weglassen?
Sind zu wenige Teilprojekte vorhanden: Welche kann man aufteilen, welche würden gut dazu passen?
Teilprojektzahl mehrmals erhöhen und absenken kann Sturktur verbessern

Gesamtprojekt mit wenigen Worten zusammenfassen: Worum geht es? Optimal: Ein Satz
Kann sowohl für die Projektplanung, aber auch für die Projektvorstellung verwendet werden

Kritische Teile

Projektteile aufschreiben und anordnen

Welche Teile müssen fertig gestellt werden, damit Projekt insgesamt funktioniert? Gibt es Teile, die auf jeden Fall drin sein müssen, gelten diese als kritisch

Kritische Teile mit hohem Aufwand oder hoher Fehlerwahrscheinlichkeit bringen das Projekt ziemlich sicher zum Scheitern

Weniger Kritische Teile

Je weniger kritische Teile, desto höher die Erfolgswahrscheinlichkeit

Techniken zum Reduzieren kritischer Teile

- Alternativen überlegen für komplexe Teile
- Projektumfang reduzieren
- Teile nicht als kritisch markieren, nur weil man sie möchte

Einer der Gründe für Modsysteme: Besser kombinierbare unabhängige Teile, reduziert auch Anzahl kritischer Teile

Projekte retten

Projektplan im Projektlauf immer wieder anpassen: Welche Teile geschafft, wie viel Aufwand übrig, wie ist aktuelle Gesamtaufwandschätzung?

Wenn sich das Projekt verzögert: Prüfen, ob Plan noch realistisch? Welche Teile stehen im Weg? Was kann man weglassen? Projektziele ändern, bis Projekt wieder durchführbar

Keine Scheu Teile wegzuwerfen, die mal viel Aufwand gemacht haben; waren trotzdem nicht unnütz, haben den Prozess weitergebracht

Fehlerkosten

Software enthält fast immer Fehler. Wie damit umgehen?

Fehlermanagementplan, für jeden offenen Fehler einschätzen

- Wie groß ist Aufwand zum Beheben?
- Welche Auswirkungen hat der Fehler, wenn er drin bleibt?
- Gibt es Alternativen zum Fehler beheben?

Dann Fehlerbehebung nach Kosten/Nutzen-Verhältnis priorisieren

Ziel sollte fehlerfreie Software sein, gibt aber auch Beweggründe dagegen: Besser Produkt mit Fehlern, als kein Produkt; Aufand zum Beheben nicht wert (Fehler tritt zu selten auf, hat zu geringe Auswirkungen, . . .)

Fertige Projekte verbessern

Übrige Zeit nach Fertigstellung kann produktiv genutzt werden

- Sind nicht-kritische Teile übrig, die noch eingebaut werden können
- Testspieler benchmarken: Wie viel Zeit wird an welchen Stellen verbracht (wie lang ist welche Graphik zu sehen, wie viel Zeit wird mit welchen Teilen der Spielmechanik verbracht, . . .)
  An Stellen, an denen viel Zeit verbracht wird optimieren: Graphiken verbessern, Texte optimieren, alternative Inhalte anbieten, . . .