Alterações

**Validação:**

Separado em:

* *Especificação*: especificação de requisitos do sistema desejados pelo cliente.
* *Validação*: validação das especificações com o cliente para verificar se o que foi especificado esta de acordo com o desejado.

A parte do modelo já validado pode sofrer alterações em seu índice de validação de acordo com o tempo passado, e dependendo da volatilidade dos requisitos. Quando maior o tempo entre as sessões de validação do cliente, maior a taxa de redução.

Fatores que contribuem na validação:

* *Modelo do sistema:* É possível fazer a validação através do modelo criado pelo analista, porém por este modo existe um limite em que a validação pode chegar (80%).
* *Protótipos*: Este método de validação pode ultrapassar o limite anterior. Os protótipos são desenvolvidos pelo arquiteto.

**Bugs:**

Dividido em quatro categorias (tipos):

* *Unitário*: Encontrado pelo testador e corrigido pelo programador.
* *Integração*: Encontrado pelo testador e corrigido pelo programador.
* *Sistema*: Encontrado pelo testador e corrigido pelo programador.
* *Aceitação*: Este tipo de bug é encontrado durante as interações com o cliente pelo analista. O analista também é responsável em desenvolver os casos de testes para este tipo.

Alem disso, o numero total de bugs no sistema não é mais visível ao jogador. Será apenas mostrado o numero de bugs encontrado pelo testador e o numero de bugs corrigidos.

**Testador:**

A função do testador foi alterada. Agora ele apenas encontra os bugs no sistema. Quem corrige os bugs são os programadores.

**Programador:**

Além de codificar, o programador agora tem a função de remover os bugs do sistema. Para uma quantidade X de bugs removidos, existe a chance de adicionar um novo bug (razão de 5 para 1).

**Qualidade de código:**

Afeta a arqiotetira e influencia nas probabilidades de corrigir bugs e gerar novos bugs por consequencia, afetando a razão de 5 para 1 (5 bugs corrigidos, 1 inserido).

**Gerente:**

É responsável em escalonar os papeis de cada membro da equipe e pela contratação.

*Contratação:*

Pode ser realizado de duas maneiras:

* *Longo Prazo*: O gerente avalia o potencial de cada candidato.
* *Curto Prazo*: Avalia quem é o mais qualificado para as especificações do projeto(especializações).

Fatores que contribuem para contratação e para o desenvolvimento:

* *Tecnologia*: Este fator se refera as especializações do funcionário.
* *Domínio do software*: Se refere à experiência que o funcionário tem em trabalhar em diferentes domínios de softwares. (Projetos já trabalhados)
* *Familiaridade com a equipe*: Quanto tempo que está na equipe.
* *Conhecimento do projeto*: Quanto tempo exerceu cada papel naquele projeto.

**Papeis:**

*Papel Secundário:*

O jogador pode decidir se os funcionários exercerão papeis secundários, onde para cada funcionário, o jogador escolhe qual vai ser o nível de dedicação do papel primário e secundário (60-40%, 70-30%, 80-20%).

Decisões:

*Programador*:

Duas decisões chaves:

* *Evolução*: Gera linhas de código e insere bugs.
* *Reparar*: Corrige bugs no sistema e insere bugs colaterais da correção.

Como exercer sua tarefa:

* *Gambiarra*: Maior produtividade, maior penalidade, piora qualidade do código.
* *Desenhar-Codificar*: intermediário (Default).
* *Test-Driven*: Menor produtividade, mas menor penalidade.
* *Gerar documentação*: Menor produtividade, menor penalidade, auxilia "newbies".
* *Refatoração*: Menor produtividade, melhora qualidade do código.

*Analista*:

Decisões:

* Comunicação intensa com o cliente: Valida o modelo proposto e ajusta o modelo, podendo ter "perdas" no modelo. Cliente informa bugs caso encontre.
* Especificação de Requisitos: Elicitação de requisitos, mas não valida.
* Gerar casos de testes de aceitação.
* Gerar "Contratos de Operação": Gera modelo, em uma produtividade reduzida, mas auxilia "newbies".

*Arquiteto*:

Decisões:

* Criar protótipo: Usado para auxiliar o analista durante a validação do modelo/software.
* Gerar casos de teste de sistema
* Gerar casos de teste de integração
* Trabalhar na arquitetura/modularizar o código: auxilia os programadores da equipe.
* Gerar documentação da arquitetura: Modo (ativado ou não) que reduz a produtividade, mas auxilia "newbies".

*Testadores*:

Decisões de estado, afetados pela estamina e moral:

* Ótimo: Produz mais
* Normal: Default
* Ruim: Produz menos

*Gerente*:

Decisões:

* Determinar o foco da iteração
  + Produtividade
  + Analise
  + Testes
  + Default
* Contratação
* Escalonar funcionários
* Estimativas:
  + Planning Poker: Rapido, impreciso
  + Cocomo: Demorado, "preciso"

*Gerente de riscos*:

Decisões:

* Identificar riscos
* Monitoramento de riscos
* Mitigação de riscos

**Treinamento:**

Dividido em tipos:

* Especialização.
* Familiaridade com o papel.
* Familiaridade com o projeto.

**Modos de jogo:**

Escolhe-se que tipo de cenário deseja praticar. O tipo de cenário selecionado afeta o nível de dificuldade correspondente nos projetos.

* Codificação
* Analise
* Testes

**Interessante:**

* Diagrama de atividades
* Diagrama de fluxo de dados
* Escolher o modelo usado para arquitetura
* WBS
* Expandir metodologias: Dividir em métodos usuais (Scrum, Cascata, etc)