**SURVEY/GRUPO FOCAL – Planejamento:**  
**1 — Capacidade e escolha do tamanho da amostra (decisão óbvia)**

Capacidade total de avaliações = 15 devs × 10 métodos/dev = **150 avaliações**.  
Cada método precisa de 5 avaliações → número máximo de métodos = 150 / 5 = **30 métodos**.

**Portanto: selecionaremos 30 métodos para avaliação primária.**  
Além disso, vamos gerar uma **lista de reserva** com 10% a 20% a mais (3–6 métodos extras) para substituir métodos excluídos por problemas ou conflito de interesses — estes *reservas* não entram na rodada inicial, só se necessário.

**2 — Estratégia de amostragem (estratificada e por risco/complexidade)**

Objetivo: garantir que os 30 métodos reflitam a diversidade do backend (camadas, complexidade, presença de exceções, existência/ausência de documentação).

**Estratos sugeridos (exemplos a aplicar com seus metadados):**

1. Camada arquitetural — 8 camadas (config, exception, model, report, repository, resource, service, util).
2. Complexidade do método — baixa / média / alta (usar métricas: linhas de método e/ou complexidade ciclomática).
3. Presença de exceções — métodos com throw/catch / sem exceções.
4. Presença de documentação original — sem Javadoc / com Javadoc parcial / com Javadoc completo.

**Procedimento de alocação estratificada:**

1. Abaixo segue como calculamos o número de métodos por camada no *parque* inicial (dos ~200 métodos). Como já temos dados de contagem por camada (Summary.docx), usaremos para alocar proporcionalmente:
2. Dentro de cada camada, sub-estratifique por complexidade (por exemplo: 40% baixa, 40% média, 20% alta), ou faremos a alocação proporcional ao número real em cada subgrupo.
3. Se um estrato pequeno der zero por arredondamento, asseguraremos pelo menos 1 método (quando possível) para cobertura mínima.

Caso não tenhamos counts por camada: estratificaremos basicamente por *camada* com alocação igualitária inicial (30 / 8 ≈ 3–4 métodos por camada) e depois ajustaremos por relevância (service e model tendem a merecer mais amostras).

**3 — Critérios de ranking / seleção dentro de cada estrato**

Dentro de cada estrato, ordenaremos os métodos seguindo uma pontuação composta (score de prioridade), para escolher os top-k de cada estrato:

**Score Prioridade = w1·(1 − doc\_presence) + w2·norm(LOC) + w3·norm(cyclomatic) + w4·exception\_flag**

* doc\_presence: 0 = sem Javadoc, 0.5 = parcial, 1 = completo → queremos priorizar menor documentação (1 − doc\_presence).
* LOC: linhas de código do método (normalizado 0–1).
* cyclomatic: complexidade ciclomática (normalizada 0–1).
* exception\_flag: 1 se faz/lança exceção, 0 caso contrário.
* pesos sugeridos: w1=0.35, w2=0.25, w3=0.25, w4=0.15 (ajuste conforme prioridades).

Escolha os **top N** por estrato segundo o ScorePrioridade até encher a cota do estrato.

**4 — Geração da lista final (30 + reservas) — passos práticos**

1. Obtenha metadados para os ~200 métodos: camada, LOC, cyclomatic (use ferramenta estática: e.g. javancss, pmccabe, SonarQube), presença de Javadoc, presença de throw/catch.
2. Calcule estratos e alocação proporcional (ver passo 2).
3. Calcule ScorePrioridade dentro de cada estrato e selecione os top k estrato.
4. Monte a lista final de **30 métodos** e **4 reservas** (por ex. 10% reservas = 3 → use 4 para margem).
5. Gere um arquivo CSV com colunas: method\_id, arquivo, classe, camada, LOC, cyclomatic, doc\_presence, exception\_flag, score, selected(yes/no), reserve(yes/no).

**5 — Plano de alocação de avaliações (balanceado e determinístico)**

Com 30 métodos e 15 avaliadores, cada avaliador fará exatamente 10 métodos (pois 30×5 =150 e 150/15=10). Isso facilita balanceamento perfeito.

**Algoritmo simples e reprodutível (pseudo-procedimento):**

1. Numere os métodos 1..30.
2. Liste os avaliadores D1..D15.
3. Para r = 1..5 (cada método precisa de 5 avaliações):

Gere uma **permutação aleatória** dos avaliadores com uma semente (seed) fixa para reprodutibilidade, por exemplo perm\_r.

Associe os métodos a avaliadores por rodada r em chunk de tamanho ceil(30/15)=2 mas devemos garantir cada avaliador receba ao final 10; procedimento prático abaixo garante equilíbrio:

1. Procedimento prático e balanceado (forma direta):

Crie uma matriz vazia Assign[Dev][Method] = 0.

Repetir até cada método ter 5 atribuições:

* + - Shuffle developers list (seeded).
    - Itere métodos do 1..30; para cada método, atribua ao próximo developer da lista que ainda tem <10 métodos atribuídos e que ainda não foi atribuído a esse método (evita duplicatas).

Este looping termina exatamente quando 150 atribuições são feitas; com cuidado a restrição de 10 por dev será satisfeita porque 15×10=150.

1. Grave o resultado em planilha com colunas method\_id, dev\_assigned.

Se preferir, pode aplicar um **design de blocos incompletos balanceados (BIBD)** matematicamente perfeito — aqui não é necessário, dado que números batem perfeitamente.

**6 — Procedimentos operacionais e logísticos**

* **Seed e reprodutibilidade:** use uma seed única (ex.: 20251009\_Murillo) para todas as rand() — armazene seed no CSV.
* **Conflictos / conflito de interesse:** marque desenvolvedores que trabalharam nos métodos; evite atribuí-los aos mesmos métodos (ou marque como observador).
* **Tempo por avaliação:** estime ~15–25 minutos por método (depende da complexidade). Com 10 métodos ≈ 2.5–4.5 horas por dev → combine com disponibilidade.
* **Instruções para avaliadores:** entregue um kit com: método original, método Amazon Q modificado, Markdown gerado, log de exceções, formulário de survey e prazo.
* **Reservas:** se algum dev recusar, reatribua conforme lista de reserva; reserve 1–2 devs como substitutos.

**7 — Survey / Instrumento de coleta (mínimo recomendado)**

Cada avaliação (por método) deverá coletar:

**Seção A — Identificação**

* method\_id; avaliador\_id; data; tempo gasto (min).

**Seção B — CHT (Likert 1–5)**

* C1: A documentação contém descrição geral adequada? (1=péssimo … 5=excelente)
* C2: Os @param estão corretos e completos?
* C3: O @return está adequado ou presente quando necessário?
* C4: As exceções listadas (@throws) são coerentes com o código?
* H1: A documentação é clara e legível?
* H2: A documentação ajuda a entender o propósito do método (utilidade prática)?
* H3: Você confiaria nesta documentação para implementar mudanças seguras?
* T1: As entidades e referências citadas existem no repositório e são acuradas? (1–5)
* T2: Há alguma informação incorreta ou “alucinada”? (sim/não) e se sim, descreva.

**Seção C — Comentários abertos**

* Pontos fortes (texto curto).
* Pontos fracos / correções sugeridas (texto curto).
* Classificação final do método (aceitar/pedir revisão/rejeitar).

**Seção D — Checklist binário (rápido)**

* @param presentes (S/N), @return presente (S/N), @throws presente (S/N), exceções reais confirmadas (S/N).

**8 — Análise estatística pós-coleta**

1. **Cálculos principais:**

Média e desvio-padrão de C-Score, H-Score, T-Score por método.

Agregados por camada (média por camada) e por complexidade.

QDI por método e por camada.

1. **Confiabilidade inter-avaliadores:**

Use **Fleiss' Kappa** (múltiplos avaliadores categóricos) ou **Intraclass Correlation (ICC)** para escalas contínuas/Likert.

Compute **Cronbach’s alpha** para consistência interna das perguntas Helpfulness se forem somadas em uma escala.

1. **Testes de hipótese (se necessário):**

ANOVA ou Kruskal-Wallis para comparar scores entre camadas ou grupos de complexidade.

Correlação (Spearman) entre complexidade (LOC / cyclomatic) e ganho de completude (delta C).

1. **Detecção de outliers e problemas de avaliação:**

Verifique avaliadores com média muito diferente → possível viés.

Compare tempo gasto vs. qualidade percebida para detectar respostas apressadas.

1. **Relatórios e visualizações:**

Boxplots por camada (C/H/T).

Heatmap de QDI por camada × complexidade.

Tabelas com top-10 métodos com maior melhoria e top-10 com maior risco (alucinações).

**9 — Indicadores de aceitação / critérios para adoção**

Sugestões de limiares práticos:

* **Aceitar documentação automaticamente** se: C-score ≥ 0.9, H-score ≥ 4.0, T-score ≥ 0.95.
* **Revisão humana obrigatória** se: T-score < 0.9 ou qualquer avaliador marcar “informação incorreta” (T2=sim).
* **Rejeitar / reprocessar com few-shot** se: C-score < 0.6 ou H-score < 3.0.

(Ajuste limites conforme necessidades institucionais.)

**10 — Plano de entrega e artefatos**

Você deve obter como artefatos finais:

1. CSV com 30 métodos selecionados + reservas + metadados.
2. Planilha de atribuições (who evaluates what) — gerada pelo algoritmo de alocação.
3. Pacote para cada avaliador: originais + Amazon Q outputs + formulário online (Google Forms / LimeSurvey / Qualtrics).
4. Base de dados com respostas (CSV/Excel).
5. Script(s) para cálculo de C/H/T e QDI (pode fornecer notebook em Python se desejar).
6. Relatório estatístico (médias, IC95%, testes, kappa/ICC).

**contexto do experimento** (Amazon Q → ChatGPT → Amazon Q → CHT → survey com 15 desenvolvedores), podemos desenhar um **instrumento completo de coleta para o grupo focal**.

Esse instrumento deve combinar:  
perguntas quantitativas (para comparação entre métodos e estatísticas CHT)  
perguntas qualitativas e de reflexão (para capturar percepções sobre uso do Amazon Q e do processo)

**Estrutura sugerida do instrumento do grupo focal**

**Título:**  
*Avaliação da Documentação Gerada pelo Amazon Q no Sistema SASB – Grupo Focal de Desenvolvedores*

**Objetivo do instrumento:**  
Coletar percepções técnicas e qualitativas sobre a documentação gerada automaticamente pelo Amazon Q (AWS) em métodos Java do backend SASB, complementando os indicadores do framework CHT (*Completeness*, *Helpfulness*, *Truthfulness*).

**Seção 1 – Perfil do Participante**

1. Função atual no projeto ( ) Analista de sistemas ( ) Desenvolvedor ( ) Testador ( ) Outro: \_\_\_\_\_\_
2. Tempo de experiência em desenvolvimento Java:  
    ( ) < 2 anos ( ) 2–5 anos ( ) 5–10 anos ( ) > 10 anos
3. Grau de familiaridade com o SASB:  
    ( ) Nenhuma ( ) Baixa ( ) Média ( ) Alta
4. Já utilizou ferramentas de IA (ChatGPT, Amazon Q, Copilot etc.) para gerar documentação?  
    ( ) Sim ( ) Não
5. Caso sim, qual(is)? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Seção 2 – Avaliação Individual dos Métodos (por método)**

**Instrução:**  
Para cada método que você recebeu (máx. 10), avalie a documentação gerada pelo Amazon Q conforme os critérios abaixo, comparando o **código original** e o **código documentado**.  
Escala: 1 = Discordo Totalmente  5 = Concordo Totalmente

**A. Completeness (Completude)**

1. A descrição geral do método é adequada e cobre seu propósito principal. (1–5)
2. Todos os parâmetros estão descritos corretamente nos @param. (1–5)
3. O retorno do método está claramente documentado em @return. (1–5)
4. As exceções (@throws) estão listadas e descritas corretamente. (1–5)
5. O formato geral do Javadoc segue o padrão Java e é consistente. (1–5)

**B. Helpfulness (Utilidade)**

1. A documentação é clara e fácil de entender. (1–5)
2. As explicações ajudam a compreender rapidamente o propósito do método. (1–5)
3. A documentação seria útil para alguém que não conhece o código. (1–5)
4. O estilo e a linguagem estão adequados ao padrão técnico da equipe. (1–5)
5. As instruções fornecidas permitem reuso ou manutenção segura do método. (1–5)

**C. Truthfulness (Veracidade)**

1. Todas as informações apresentadas correspondem fielmente ao código. (1–5)
2. Nenhum comportamento inexistente foi descrito (“alucinação”). (1–5)
3. Os nomes de variáveis, classes e exceções citados realmente existem. (1–5)
4. O conteúdo do Markdown reflete a lógica real do método. (1–5)

**D. Síntese de Avaliação**

1. Avaliação geral da documentação gerada pelo Amazon Q: (1–5)
2. Tempo estimado gasto para avaliar este método: \_\_\_ minutos

**E. Observações abertas (por método)**

1. Pontos fortes desta documentação: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Pontos fracos / melhorias sugeridas: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Você confiaria nesta documentação para manutenção futura? ( ) Sim ( ) Não

**Seção 3 – Discussão coletiva (grupo focal)**

**Estas perguntas são para o momento presencial ou remoto de debate coletivo (gravado e transcrito).**

**Tema 1 – Percepção geral do Amazon Q**

1. Quais foram suas primeiras impressões sobre a qualidade da documentação gerada automaticamente?
2. Em comparação com a escrita manual, como avalia o equilíbrio entre velocidade e precisão?
3. O modelo captou corretamente o contexto funcional dos métodos?

**Tema 2 – Interação IA–Desenvolvedor**  
4. Em que situações o Amazon Q poupou tempo de entendimento do código?  
5. Em quais casos ele introduziu ambiguidades ou erros?  
6. Você confiaria na IA para documentar módulos críticos do SASB? Por quê?

**Tema 3 – Impacto e usabilidade**  
7. Que tipos de métodos ou camadas do sistema parecem se beneficiar mais da documentação automatizada?  
8. Que melhorias você sugeriria no processo (prompt, formato, revisão humana)?  
9. Como o uso combinado Amazon Q + ChatGPT afetou a qualidade final?

**Tema 4 – Aceitação e ética**  
10. Há preocupações quanto à autoria, responsabilidade ou confiabilidade da IA na documentação técnica?  
11. Que política interna deveria reger o uso dessas ferramentas na ANA/SASB?

**Seção 4 – Síntese pós-grupo (questionário final individual)**

1. O processo iterativo (Amazon Q → ChatGPT → Amazon Q) trouxe melhoria perceptível na qualidade? (1–5)
2. Você acredita que ferramentas como o Amazon Q podem reduzir a dívida técnica de documentação? (1–5)
3. Classifique o esforço cognitivo necessário para revisar a documentação gerada: (1–5)
4. Qual seria o nível de aceitação institucional aceitável (em %) para adotar documentação automática sem revisão humana? \_\_\_ %
5. Comentário final ou insight relevante: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Resultados esperados e análise**

* As perguntas **1–15** alimentam os indicadores **C, H, T** do *Evaluation Framework* (Meta).
* As perguntas **16–19** e as de grupo focal geram **insights qualitativos** (codificação aberta e categorização temática).
* As de síntese permitem medir **aceitação tecnológica (TAM)** e **viabilidade de adoção**.