Software Engineering Introduction 2018 想定問題

1. ソフトウェア工学とは何かを授業内容を用いて説明せよ
   ソフトウェア工学とは、ソフトウェアを通した課題解決において一定の品質を確保するための技術、道具やパラダイムなどを含む知恵のこと。
2. ソフトウェア開発の基本プロセスを答えよ
   要求定義→設計→実装→テスト→運用
3. 要求定義とは何か説明せよ
   要求定義とは、注文主が依頼するソフトウェアに必要としている動作や機能を依頼者の言葉で定義したもの。
4. 要求定義の種類を 4 つ答えよ
   機能要求・品質要求・デザイン要件・プロセス要件
5. ユースケースを考える目的を説明せよ
   動作や機能を明らかにすること
   関与する人や物を明らかにすること
6. 「モデル化」することの役割とそれによって得られるメリットを答えよ
   要求をまんべんなく理解し、未定義や曖昧、要求の矛盾に気づくことがモデル化の役割で、依頼者と開発者の確実な共通理解を実現できることがモデル化によって得られるメリットである。
7. 要求モデルの主要なパラダイムを 4 つと、それぞれの特徴、具体的な成果物例を答えよ
   データ・フロー図: 機能間のデータの流れを表す。直感的。機能とデータの依存関係が分かる。曖昧なのでイライラする。UML ユースケース図が例。
   ER 図: エンティティ、関係、属性で構成され、概念モデルを記述する。概要を理解しやすい。要求が変わっても見た目があまり変わらない。エンティティと属性の区別が難しい。適当なレベルでのモデル化が難しい。UML クラス図が例。
   イベント・トレース: 一連のエンティティ間イベントのやり取りを時間軸に沿って記述する。UML シーケンス図が例。
   ステートマシン図: イベント間の状態やイベント発生時のふるまいを表す。動的なふるまいを記述できる。一連のイベントに対する振る舞いの変化を記述できる。システムが認識すべき入力と出力を定義できる。状態を抜き出して選ぶ必要がある。
8. UML ユースケース図の役割を答えよ
   プロジェクト初期にトップレベル機能を記述。重要なエンティティを特定する。
9. UML クラス図の役割を答えよ
   重要なエンティティとその関係を記述。他の図の理解から詳細化する。
10. UML シーケンス図の役割を答えよ
    複数オブジェクト間の重要なシナリオを記述。共通のサブシーケンスを見つけて状態を明らかにする。
11. UML コラボレーション図の役割を答えよ
    複数のイベント・トレースをクラス図の上に記述。メッセージを時系列に記述してクラス間の関係を詳細化する。
12. UML ステートマシン図の役割を答えよ
    クラスごとの振る舞いを記述。詳細な情報を必要とするので要求工程の最後に作る。
13. SDL システム図の概要を説明せよ
    トップレベルのブロックと通信路を記述。信号の種類をラベルに記載。

14. SDL ブロック図の概要を説明せよ

ブロックと通信路をより詳細に記述。

15. SDL プロセス図の概要を説明せよ

状態の繊維を表すステートマシン。遷移は入力や判定、タスク出力などの一連の要素であらわされる。

16. 要求定義のレベルを均一化するための指針にはどんなものがあるか答えよ

ひとつの項目に要求はひとつだけ。一つの要求から他の要求への参照は避ける。似た要求をまとめる。

17. 「システムが実現するふるまい，システム外環境のふるまい」などのシステムに関係する振る舞いをなんというか

ドメイン知識。想定条件。

18. 要求工程のプロセスを答えよ

要求→設計→実装→テスト→ドキュメント

19. 開発工程のプロセスを答えよ

要求→分析→設計→実装→検証

20. 要求の妥当性確認において基準となる要求の性質を 7 つ答え、それらについて具体的に説明しなさい

正当性…要求が要求の理解と合致するか

無矛盾性…矛盾する要求や条件がないか

非曖昧性…要求を読んだ人たちの解釈は一致するか

完全性…すべての状態や条件におけるすべての入力に対する出力、振る舞いが定義されているか

実現可能性…注文主の要求にこたえられる解決策が存在するか

検証可能性…成果物が要求を満たすことを確認する手段はあるか

追跡可能性…各要求が簡単に参照できるようになっているか、定義書と仕様書の要求は一致するか

21. 妥当性確認の方法としてあげられるウォークスルーとインスペクションについて説明しなさい

ウォークスルー…開発チームで行う非公式に開発チームで欠陥の発見を目指して行われる

インスペクション…開発チーム外の第三者が同席し予め用意したチェックリストに従って欠陥を検証する

22. 妥当性確認(Validation)と検証(Verification)の定義を述べ、またそれらの違いについて述べなさい

妥当性確認…求められているシステムを開発していることを確認。注文主の要求をすべて満たしているかどうかを確認。

検証…システムを間違いなく開発するための確認。奨励される設計指針に従っているか確認。

妥当性確認は、開発している成果物が要求に合致しているかどうかを確認するもので、検証は成果物が間違いなく完成するかどうかの確認をするものである。

23. 「要求記述言語」とは何かを説明しなさい

要求の性質、酒類に応じどのモデルを使えばよいかの統一的な指針。課題と開発チームの経験で使い分ける。

24. 要求定義と要求仕様の違いを説明せよ

要求定義は依頼主が成果物に必要とする機能や要件といった要求を依頼者の言葉で書いたものである一方、要求仕様は成果物に実装する要求を開発者の言葉で再定義したものである。

25. 設計に既存の解決策を活用する方法としてあげられる「複製」と「参照モデル」の違いについて述べよ

「複製」は既存の設計、またはコードをほぼそのまま使用することで、「参照モデル」はアプリケーションの分野ごとに汎用的なアーキテクチャを当てはめること。

26. 「アーキテクチャ・スタイル」とは何か説明し、その例を 3 つとその例に対する説明を述べよ

アーキテクチャ・スタイルとは分野に依存しないソフトウェア・アーキテクチャで、複数のスタイルを

組み合わせることでソフトウェアを実現する。アーキテクチャの選択肢を提案するガイドライン。

・デザインパターン…汎用性の高い解決策を示す。低レベル設計に対する具体的解決策。

・デザインコンベンション…一定の価値基準に基づいた設計や指針のアドバイス。従うべき基準と設計の方法を示す。

・デザインプリンシプル…良い設計の基準を示し、どのように設計するかまでは決めない。既存の解決策でなく新しい設計を生み出すのに適する。

27. アーキテクチャの設計手法として、分解手法を用いた設計における 5 つの主な分解指向とそれらの説明をしなさい

機能分解…機能や要求をモジュール化

プロセス指向分解…プロセスをモジュール化

イベント指向分解…イベントをモジュール化

データ指向分解…データの単位でモジュール化

オブジェクト指向分解…オブジェクトをモジュール化

28. Pipe-Filter の特徴を 4 つ、デメリットを 2 つ挙げなさい

・システムをフィルターの重ね合わせとして考える

・それぞれのフィルターは独立しているので再利用が可能

・システムの保守や機能的拡張が容易

・並列実行がしやすい

29. Client-Server の構成要素を 2 つ、具体的な要求・応答プロトコル、この様式を活用しているサービスの例を 1 つ挙げ、またコールバックとは何か答えよ

サービスを提供する”Server”とサービスを利用する”Client”から構成される。

Client が要求して Server が応える。

AINS が例。

コールバックとは Server の準備が完了した時点で Client を呼び出すこと。

30. Peer-to-Peer の特徴を 3 つ、そのコンポーネントの性質を 2 つ、この様式を利用したサービスの例を 3 つ、またどんなときに Peer-to-Peer よりも Client-Server が優れているといえるか述べよ

・高い処理能力のシステムを実現しやすい

・システムの拡張が容易

・耐障害性が高い

・それぞれのコンポーネントは Client にも Server にもなり得る

・すべてのコンポーネントが他のコンポーネントに要求を出せる

ただしコンテンツが頻繁に更新されたりアクセスの高速化や品質の確保、信頼性が必要とされたりする場合は Client-Server の方が有利。

Winny, Napster, Bittorrent が例。

31. Publish-Subscribe について説明し、この様式の特性とデメリットを 2 つずつ、またサービスの例を 1 つ挙げなさい

ブロードキャストとイベント応答でコンポーネント同士がやり取りする様式。

・システムの拡張が容易

・再利用性に優れている

・共有メモリが必要なためデータ通信が難しい

・発信側は受信側の事情を知らないため、妥当性検証が難しい。

Java のイベントリスナーが例。

32. Repository の構成要素を 2 つ、「データベース型」と「黒板型」のそれぞれの概要を述べよ

中央データとそれを操作するコンポーネントで構成される。

データベース型: 中央データがデータベースの役割を果たし、利用するコンポーネントは能動的に動作する。

黒板型: 中央データはサーバーの役割を果たし、コンポーネントは受動的に動作する。

・高い公開性

・中央管理がしやすい

・広く利用しやすいデータ構造を維持するのが難しい

ウィキペディアが例。

33. Layering の特徴を 2 つ、デメリットを 2 つ、またこの中の各層の性質を答えなさい

・複雑な問題を階層に分割するため、抽象度での整理がしやすい

・依存関係が前後の層に限られるため、拡張が容易

・階層構造に整理するのが難しい

・層間の通信が負荷や遅延につながるため、パフォーマンスに不安

34. ソフトウェア設計プロセスの最終成果物を答えよ

ソフトウェア基本設計書（Software Architecture Definition）

35. アーキテクチャ評価の品質観点を 7 つ答えなさい

・更新性

・性能

・セキュリティ

・信頼性

・堅牢性

・ユーザビリティ

・ビジネス価値

36. KWIC 問題とは何か答えよ

Key Word in Context 問題のことであり、求められるソリューションを実現するのに異なるアーキテクチャを用いることができること。

37. 抽象化の目的と利用方法について説明せよ

抽象化することでデータや他の機能の変更に左右されにくくし、モジュールの再利用を容易にする。

複数のクラスに共通する機能や要素を抜き出して抽象度を上げたクラスに追加しそれを継承させたり共通のモジュールとして合成させたりする。

38. アーキテクチャの評価方法を 3 つ答えよ

品質測定…メトリックスなどの数値で結合度や強度を数値化して評価する。

トレードオフ分析…複数の観点からアーキテクチャを比較し妥当性をもって評価する。

費用対効果分析…コストを予測して比較する

39. モジュール設計の原則のうちモジュール性について、結合度(coupling)と強度(cohesion)の違いを述べよ

結合度はモジュールとモジュールの間の依存性を表し、この結合が弱いほど他のモジュールの変更に左右されない良い設計といえる。

強度はひとつのモジュール内部のオブジェクト同士の関連性を表し、この関連が強いほど良い設計といえる。

40. 主な結合の段階を 5 つ述べ、それぞれについて説明しなさい

内容結合…他のオブジェクトの内部を参照している

共有結合…グローバルデータに依存している

制御結合…処理が引数の種類やパターンに依存して変化する

スタンプ結合…構造体のデータ型に依存する

データ結合…単純なデータ型に依存する

41. 主な強度のレベルを 7 つ答えなさい

偶発的…パーツ間に関連性がない

論理的…同じ論理構造を共有しているだけ

時間的…同じタイミングで使われるだけ

手続き的…時間に加え関連した目的やアクションを持つ

通信的…同じデータに対する操作をし、入出力で関連づいている

機能的…一つの要素に必要な要素がまとまっている

情報的…機能的な強度をオブジェクト指向やデータ抽象化で実装した

42. 主要なモジュール設計の原則を 6 つ答えよ

・モジュール性

・インターフェース

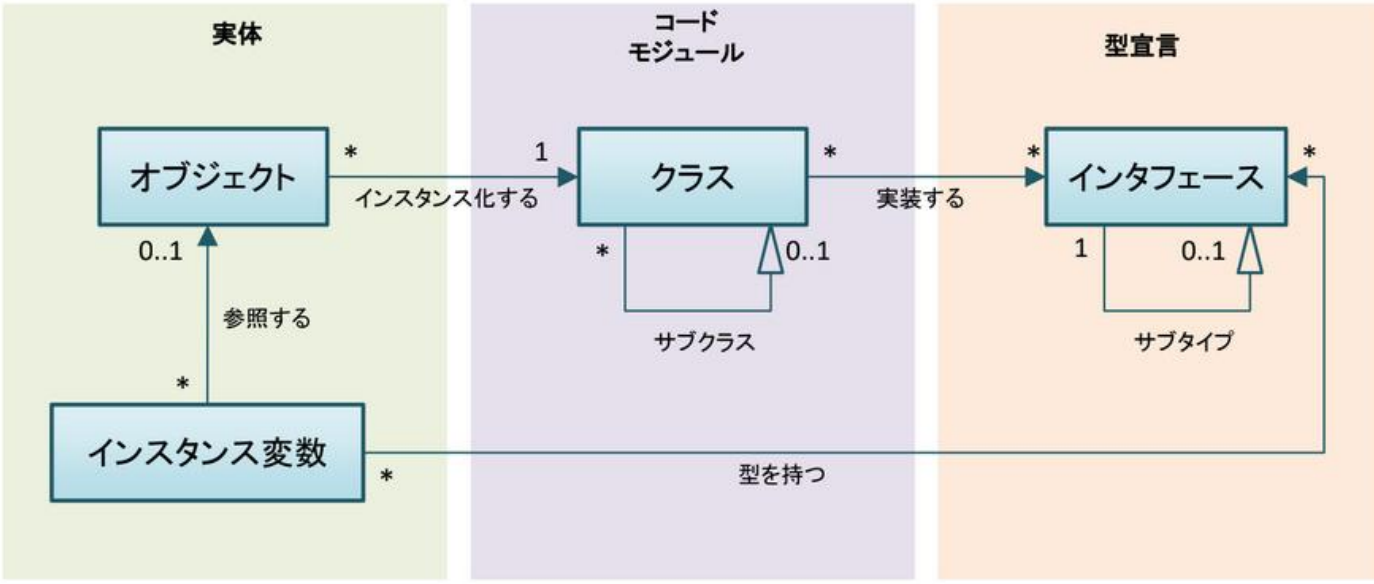
・情報隠蔽

・インクリメント開発

・抽象化

・一般化

43. オブジェクト指向プログラミングにおける 4 大要素の関係について以下の図の空欄を埋めよ



44. 多態性(polymorphism)とは何か説明せよ

動的に動作を変更することができる性質で、実行時に関連付けられるオブジェクトとメソッドにより動作が決定する。オブジェクト指向の強力な特徴。

45. 「継承」と「合成」の定義とそれらの利点と欠点を答えよ

継承…既存のクラスの振る舞いを拡張したり上書きしたりすること

・継承されたメソッドの動作を変更・特化できる

・動作を予測しやすく理解しやすい

・サブクラスの実装は設計時に決定される

合成…小さなクラスを組み合わせて複合クラスを作ること

・再利用するコードはカプセル化しやすい

・組み合わされたクラスはインターフェースを通してアクセスする

・実行時の条件に応じて適合性のあるオブジェクトで置き換えが可能

46. Liskov の置換原則の概要を述べよ

サブクラスの振る舞いは親クラスの振る舞いに準拠していなければならないこと。サブクラスは親クラスのすべてのメソッドを備え、親クラスメソッドの仕様を満たさなければならない。代替オブジェクトに置き換えても問題が発生しないか確認するのに用いる。

47. デメテルの法則の概要を述べよ

合成クラスにおいて、含まれる子クラスにアクセスするメソッドを追加すること。これによってクラス間の依存性を低めることができ、エラーを含みづらく変更にも強くする。

48. 依存関係逆転の法則の概要を述べよ

具体的な実装に依存するのではなく抽象インターフェースに依存するようにすることで依存性を逆転させることができ、結合性を弱めることができる。

49. UML を用いた設計の利点を述べよ

UML はソフトウェアの設計を可視化、定義、ドキュメント化することができる。

50. デザインパターンとは何か、またそれによって得られる利点を述べよ

デザインパターンは頻繁に直面する設計上の課題に対して繰り返し使われる実績のある理想的な開放のこと。

・設計の原則にのっとった設計を促す

・理解性、拡張性、再利用性が高い設計を提供する

・開発経験の差を埋めることにつながる

51. デザインパターンの例を 4 つとそれぞれの特徴を述べよ

・Template Method パターン…同じ親クラスを持つサブクラスがある場合に重複コードを抽象クラスにまとめることでコードの重複を少なくする。

・Factory Method パターン…具体的クラス名を必要とするオブジェクト生成に必要な知識を抽象クラスのコンストラクタメソッドにまとめてカプセル化する。

・Strategy パターン…実行時にアルゴリズムの選択を可能にする。

・Observer パターン…Publish-Subscribe のアーキテクチャを実現する。

52. 設計の規模を表す要素の例を 9 つ挙げよ

・シナリオの数

・主要クラスの数

・補助クラスの数

・主要クラスごとの補助クラスの数

・サブシステムの数

・クラスの大きさ

・サブクラスが上書きする操作の数

・サブクラスが追加する操作の数

・Specialization Index = サブクラスが上書きする操作の数 * クラスのメソッド数

53. 設計の品質を表す要素の例を 6 つ挙げよ

・クラスごとのメソッドの重み

・継承の深さ

・子クラスの数
・オブジェクト間の結合性
・レスポンスの数
・メソッドの強度

54. 操作の規模を表す要素にはどんなものがあるか

平均操作サイズ
操作ごとの平均パラメータ数

55. コーディング規約によって得られる利益にはどんなものがあるか

・結合性の低減
・強度の向上
・よく定義されたインターフェースの実現
・統合テスト、メンテナンス、将来の拡張を容易にする
・全体像を分かりやすくする
・自動化ツールで一括処理できるようにする

56. コーディング規約の観点を３つ答えよ

1. 制御構造
2. アルゴリズム
3. データ構造

57. 内部ドキュメントと外部ドキュメントの違いは何か答えよ

内部ドキュメントがプログラム中のコメントやドキュメントなど、コードについての説明であるのに対し、外部ドキュメントはいつ何のためにコンポーネントが呼ばれるのか、どんなデータの流れなのかなどソフトウェアプログラムの概要に関する説明。

58. 回帰テストの概要を述べよ

修正に伴う欠陥の発生を確認することで、変更したものが変更前のものと同じ機能であることを検証する。

59. ブラックボックステストの特徴を述べ、またその主要な技法を５つ答えよ

入力に対する出力のみを確認するテスト方法で、内部構造や処理手順を考慮する必要がない一方、代表的な入出力組み合わせを試すことしかできず完全なテストが不可能。

・同値分割法…入力値を値域で分割してそれぞれの代表値でテストケースを構成
・境界値分析…領域の境界値と領域に含まれない領域近傍の値でテストケースを構成
・決定表テスト…入力や条件の組み合わせと結果の対応関係に基づいてすべての規則を網羅するようにテストケースを構成
・状態遷移テスト…ステートマシン図などの状態遷移図に基づいてテストケースを構成
・ユースケーステスト…ユースケース記述に基づいてテストケースを構成

60. ホワイトボックステストの特徴を述べ、またその主張な技法を２つ答えよ

論理パスの考慮などを経て内部構造に即したテストを行う。

・制御フローテスト…プログラムの制御構造から実行のパスを抽出してテストケースを構成
・データフローテスト…変数の宣言、利用、消滅をカバレッジ基準にしてテストケースを構成

61. テストの構成を実施される順番に並び替えよ(機能テスト・結合テスト・導入テスト・単体テスト・性能テスト・受入テスト)

単体→結合→機能→性能→受入→導入

62. コンポーネント・ドライバーとはなにか述べよ

対象のコンポーネントを呼び出し、テストケースを渡す役目のコード。

63. スタブとは何か述べよ

まだ結合しないコンポーネントの役目を模擬するためのコード。

64. 結合テストの目的は何か

設計通りに作られているかを確認

65. システムテスト（機能・性能・受入・導入テスト）の目的は何か

顧客の要求を満たすように動作することを確認

66. ボトムアップ結合の概要とその特徴を述べよ

依存性を考慮しながら下位モジュールから一段階ずつテストする

・プログラムが実行可能になるまでの時間が遅い。
・特定パスのテストが簡単
・順番の計画と制御が簡単

67. トップダウン結合の概要とその特徴を述べよ

最高位コンポーネントを、スタブを使ってテストし、段階的にスタブを実際のクラスに置き換える。

・スタブがたくさん必要
・下位の個別テストがない
・並列度がない
・特定のパスをテストするのが難しい
・順番の計画と制御も難しい

68. ビッグバン結合の概要とその特徴を述べよ

スタブとドライバーで個別にコンポーネントをテストした後、一気にすべてを結合

・原因の特定が難しい
・結合が遅い
・並列度が高い
・特定のパスのテストが容易
・順番の計画と制御が簡単

69. サンドウィッチ結合の概要とその特徴を述べよ

トップダウンとボトムアップを両方取り入れ、システムを３段階に分けてテストする。

・実行可能になるまでの時間が早い
・順番の計画と制御が難しい

70. 変形サンドウィッチ結合の概要とその特徴を述べよ

サンドウィッチ結合に加えそれぞれの結合プロセスで途中から結合するクラスに対してもテスト

・結合も実行可能になるまでの時間も早い
・並列度が高い
・パスのテストが簡単
・順番の計画と制御が難しい

71. 結合テストの手段を選ぶ際の基準は何か

コンポーネントの依存性

72. システムテストの種類を選ぶ際の基準は何か

何をテストするか

73. ソフトウェア開発のプロセス・モデルについて、ウォーターフォール・モデルの概要とその特徴を述べよ

反復のないプロセスで、各フェーズにマイルストーンがあり、それぞれのフェーズに成果物がある。

・変更が生じたときの対応が決まっていない
・最終成果物までに時間が掛かる

74. ソフトウェア開発のプロセス・モデルについて、V 字モデルの概要とその特徴を述べよ

単体テスト→プログラム設計を評価
結合テスト→システム設計を評価
受入テスト→要求を検証

75. ソフトウェア開発のプロセス・モデルについて、インクリメンタルプロセスモデルの概要とその特徴を述べよ

・短い周期で繰り返す
・システムを複数の部品に分けて納品するので問題の早期発見につながる
・リリースごとに特定の課題に集中できる

76. ソフトウェア開発のプロセス・モデルについて、プロトタイピング・モデルの概要とその特徴を述べよ

要求や設計を繰り返し確認できるので開発のリスクと不確実さを減らす

77. ソフトウェア開発のプロセス・モデルについて、アジャイル開発の概要とその特徴を述べよ

ソフトウェアを早く、確実に開発するために柔軟性を強調
プロセスやツールより人と意思の疎通を重視
文書よりソフトの生産を重視
計画遵守より変化への対応に注力

78. WBS は何の略？

Work Breakdown Structure

79. プロジェクト管理の制約を 3 つ答えよ

スコープ、時間、コスト

80. WBS の主な役割を 4 つ述べよ

・作業の網羅
・計画地の整合性
・担当の明確化
・管理指標

81. プロジェクトのコスト管理において、比較(類推)見積り、計数(パラメトリック)見積り、ボトムアップ見積りはそれぞれ何を基準に見積りするか答えよ

順番に、過去の類似の活動、過去の統計から求めた数式、WBS で分解した活動の仕事量。

82. ソフトウェア規模を見積もる際の SLOC 法、FP 法、ユースケースポイント法はそれぞれ何を基準に見積りするか答えよ

順番に、ソースコードの行数、機能要件の定量化、アクターとユースケースに重み付けした個数。

83. プロセスに従ったソフトウェア開発の利点を述べよ

誰が開発しても一定の品質と効率を確保することができる。

84. ソフトウェア工学の今までの成果と今後の課題について述べよ