**week12**

情報技術者キャリアデザイン第12週の授業を始めます。今日は情報系のさまざまな職種を紹介する回です。前回は、ITスキル標準（ITSS）やその他の普及標準の歴史的な背景についてお話ししました。

今回の授業では、ITスキル標準で定義された主要な職種について紹介します。また、ITSS+やiコンピテンシディクショナリーなど、より新しい枠組みに基づいた職種については、今後皆さんに分担して調査してもらいます。

そのため、第13週と第14週では、iコンピテンシディクショナリーに掲載されている職種をそれぞれに1つずつ割り当て、調査を行ってもらう予定です。今日の資料では、iコンピテンシディクショナリーのデータをダウンロードできるサイトを紹介しています。

具体的な調査方法については、来週・再来週の授業で説明します。今日は、ITスキル標準に基づき定義されている11の職種について紹介します。

これからスライドに基づいて説明を進めていきます。ITスキル標準においては、マーケティング、セールス、コンサルタントなど、全部で11の職種が定義されています。

これらの職種を理解するためには、ソフトウェア開発の工程を理解しておくことが重要です。理工学部では1年生にコンピュータプログラミングの授業が必修となっており、Pythonを用いた演習などが行われています。

この授業で扱うプログラミングは、ソフトウェア開発工程でいうと「詳細設計」と「コーディング」にあたります。

実際のソフトウェア開発は、「企画」「要件定義」「基本設計」「詳細設計」「コーディング」「テスト」「運用」「保守」といった複数の工程を系統立てて実施することで製品が完成します。

本授業では、「詳細設計」「コーディング」だけでなく、全体の工程についても紹介します。情報系のさまざまな職種は、それぞれが異なる工程を担当しており、その理解が職種理解につながります。

ここからは、それぞれの職種について説明します。たとえばマーケティングの職種では、顧客ニーズに応じて市場動向を予測・分析し、事業戦略や販売戦略、資金計画、チャンネル戦略、ビジネス戦略の企画・立案を行います。また、立案した戦略の投資効果や顧客満足度に責任を持ちます。

他の職種についても同様に概要が示されています。ただし、説明を読むだけでは分かりづらい部分もあるかと思います。そこで、ソフトウェア開発工程と関連付けてより具体的に解説していきます。

ソフトウェア開発においては、「企画」「要件定義」「基本設計」「詳細設計」「コーディング」「テスト」「運用」「保守」という工程を順にたどっていきます。

「企画」に関する業務は「企画プロセス」、「要件定義」に関する業務は「要件定義プロセス」と呼ばれます。すべてのソフトウェア製品はまず企画から始まり、「何を作るべきか」という方向性を定めます。

方向づけができたら、それを具体化するために要件定義を行い、仕様書を作成します。この仕様書ができると、「基本設計」「詳細設計」「コーディング」「テスト」へと移ります。これらの工程は総称して「開発プロセス」と呼ばれます。

開発が終了すれば製品は完成ですが、実務では完成後の「保守」や「運用」が極めて重要です。プログラムは作って終わりではなく、保守や運用を通じて初めて価値を生み出します。

したがって、ソフトウェア開発は「企画」「要件定義」「開発」「保守」「運用」という5つのプロセスに分類されます。今日の授業では「企画」「要件定義」「基本設計」「詳細設計」などの用語を使って説明していきます。

前回の資料でも紹介したスライドでは、ITスキル標準で定義されたセールス、コンサルタント、ITアーキテクトなどの職種が、それぞれ「経営戦略策定」「戦略的情報化企画」「開発」「運用方針」などのプロセスに従事していることが示されています。

「経営戦略の策定」では、従来人間が手作業で行っていた業務のどの部分をIT化するのかを検討します。このプロセスに関与するのがセールスやコンサルタントです。

セールスは顧客のビジョンやビジネス戦略の確認を通じて貢献します。一方、コンサルタントは、企業の目標に対して提言を行い、戦略の策定を支援します。これらの活動を通じて、ITで実現すべき業務の範囲が決まります。

その後、「戦略的情報化企画」の段階に進みます。ここではシステムの仕様策定が行われます。ビジネス課題に対するソリューションを提案し、どのような手法で戦略を実現するかを考えます。

ソリューションにはソフトウェア開発を中心としたものが多いですが、ハードウェアや情報システム全体を含むもの、あるいは業務の一部だけをIT化するものなど、さまざまな形態があります。企業のニーズに最も合った解決策を選ぶという意味で「ソリューション」と呼ばれます。

その過程で、ソフトウェアやハードウェアの仕様を決め、どのようなものを作るべきかを明らかにします。そのために、基本計画や開発計画を立てる必要があります。

仕様書が完成すれば、次は開発へと進みます。開発では、ITアーキテクトからカスタマーサービスまでさまざまな職種が関与します。ある人はコンポーネントの設計を、ある人はプロジェクト管理を、また別の人はアプリケーション開発を担当するなど、チームで役割を分担して作業を進めます。

職種によって担当業務が異なるというのがこの工程の特徴です。スライドに示されている濃い青色は「主たる活動局面」、薄い青色は「従たる活動局面」を示します。従たる活動局面では、主たる担当者を支援する役割を担います。

運用・保守の段階では、カスタマーサービスやITサービスマネジメントの職種が中心となって活動します。他の職種の人々は、主に運用支援という形で関与することになります。

さて、これが全体像ですが、順を追って説明していきます。まずは「企画」についてです。

企画とは、顧客の現状を分析し、抱えている問題点を理解する工程です。お客さんは何らかの問題を抱えており、それを解決するために情報システムを開発したり、アプリケーションソフトを導入したりします。

このように、企画の第一歩は、顧客の現状や問題点を正しく理解することにあります。顧客の抱える問題は、「ニーズ」と「ウォンツ」の2つに大別できます。

ニーズは、何らかの問題を放置すると業務に支障をきたすような、解決しなければ困るものです。一方、ウォンツは、解決すれば何らかのメリットが得られる、つまり顧客の「こうなればいいな」という欲求に応えるものです。

企画とは、ニーズやウォンツを満たすための方策を考えることです。顧客の問題を解決する可能な手段を列挙し、その中から最適なものを選びます。

どの手段を選ぶべきかは、顧客のニーズ・ウォンツを最もよく満たせるかどうか、あるいは開発側の強みを活かせるかどうか、開発コストや納期などの条件を満たせるかどうかといった観点から判断されます。

この3つの観点から最適な手段を考えるのが企画の醍醐味です。企画が好きな人は、このような問題解決の制約条件をもとに最良の方法を考えることに楽しさを見出しています。そのためには知恵や経験が必要です。

企画を担当する職種には、マーケティング、セールス、コンサルタントがあります。

マーケティング担当者は、顧客が求める製品やサービスを提供するために、企業の内部状況や市場動向を分析します。これにより、製品やサービスについて市場や消費者がどう考えているのかを把握し、現状の課題を明らかにします。

その分析結果をもとに、今後の方向性を予測し、企業が成功するための事業戦略や販売戦略、資金計画などを立案します。また、立案した戦略が投資効果、新規性、顧客満足度の向上に貢献するかどうかも評価の対象となります。これらに責任を持つのがマーケティングの役割であり、そこに面白さもあります。

セールス担当者は、マーケティングが立案した戦略に基づき、製品やサービスを顧客に提案し、契約を獲得します。彼らは実際に販売活動を行う担当であり、顧客との良好な関係を築き、満足度を高めることが重要です。

コンサルタントは、特許やノウハウなどの知的資産やコンサルティング技法を活用し、顧客の経営戦略やIT戦略に関する相談を受けて提言や助言を行います。それによって顧客の課題解決やビジョンの実現に貢献し、IT投資の経営判断を支援します。

コンサルタントは企業の外部にいて、トップと強い信頼関係を築く必要があります。そのため、幅広い知識や経験、人脈が求められます。

ここまでが企画に関する説明です。

企画が立てば、次は「要件定義」です。企画によって問題点とその解決策の方向性が定まり、それに基づいてソフトウェアに求められる機能や仕様を文書化します。

ソフトウェア業界では、仕様書が事実上の契約書として機能します。開発会社は基本的に、仕様書に書かれていることしか実装しません。書かれていないことは実装義務がないため、記載漏れがあると対応されない可能性があります。

また、記載に矛盾があると、そもそも実現不可能になります。そのため、仕様書には正確かつ明確な記述が求められます。

仕様書があれば、そこから開発コストや期間を見積もることができます。見積もりの方法もありますが、本授業では時間の都合で扱いません。興味のある人は「ソフトウェア工学」の授業を受講してください。

要件定義はプロジェクトの成否を左右する重要な工程です。情報システムプロジェクトでは、失敗するケースもあり、その主な原因は2つです。

1つは、開発者が関与せずに見積もりが行われるケースです。営業担当者が、例えば「この案件は1000万円でいけます」と提案してしまうと、その金額が一人歩きしてしまい、実際に開発者が仕様書に基づいて見積もると3000万円かかるというような事態が発生します。

そうなると赤字になってしまうため、仕様を削る必要が出てきます。このようなトラブルが、開発者が関与しない見積もりや、仕様が確定する前に金額を提示してしまうことから生じます。

もう1つは、仕様が確定しないままプロジェクトが始まってしまうケースです。納期が決まっているのに、仕様が決まらないと、十分な開発期間が確保できず、間に合わなくなります。

仕様に抜けがあることに気づかずに進めてしまうと、後から問題になりますし、矛盾する記述があると修正が難しく、仕様確定までに時間がかかります。

こうした問題を回避するためにも、要件定義の工程を円滑に進めることが非常に重要です。

仕様策定が完了すると、「基本設計」に入ります。何をどのレベルで実装するかが決まると、それに基づいてソフトウェア開発が開始されます。

基本設計はソフトウェア開発の最初のステップです。現在のソフトウェア開発は非常に大規模化しており、1000万円規模の案件でも小規模に分類されます。通常のプロジェクトは1億円〜10億円、大規模プロジェクトでは100億円〜1000億円規模になることもあります。

例えば、100億円規模のプロジェクトがあったとして、それを個人で開発することはできません。プログラムの行数から見ても何百万行という規模になり、一人で開発するのは到底不可能です。

そのため、ソフトウェアをいくつかの部分に分割し、それぞれの部分を開発者や開発チームに割り当てます。各チームが分担して部分を開発し、最後にすべてを統合してソフトウェア全体を完成させることになります。

このように分割するには、分割の方法、すなわち戦略を立てる必要があります。この分割方法には、大きく分けて二つの方針があります。

一つは、ソフトウェアの機能に基づいて分割する方法です。このように分割した単位は「モジュール」と呼ばれます。もう一つは、ソフトウェアが管理するデータに基づいてモジュールを定義する方法です。

従来は機能に基づく分割、すなわち「構造化設計」が主流でした。これは、ソフトウェアが仕様書に記された機能を実現するものであるため、自然な分割方法と考えられていたからです。

しかし、近年では機能に基づく構造化設計はうまくいかないケースが多いことが分かってきました。理由としては、開発の途中で仕様変更が発生することが多くなったためです。

このような背景から、現在ではデータに基づくモジュール設計、すなわち「オブジェクト指向設計」が主流となっています。こちらの方が、仕様変更に強いという利点があります。

構造化設計とオブジェクト指向設計は根本的に異なる考え方に基づいています。そのため、基本設計を行う際には、どちらの方針に基づいているかを明確に意識することが重要です。

いずれの設計方法においても、モジュールは作成されます。各モジュールについて、そのモジュールが提供する機能や、個々のルーチンの機能仕様を決める必要があります。

オブジェクト指向設計では、モジュールを「クラス」、ルーチンを「メソッド」と呼びます。概念的には、どちらも同じものを指しています。

このような設計作業は、ソフトウェアの基本構造を定める非常に重要なステップであり、ここで誤ると後の修正が困難になります。

基本設計が固まると、モジュールやルーチンが定義され、それぞれのルーチンの引数や戻り値、必要なデータ構造が決まります。次に、それぞれのルーチンが実現すべきアルゴリズムを決定し、それに必要なデータ構造も合わせて設計します。

データ構造とアルゴリズムは密接な関係にあります。この関係を示す有名な言葉に「アルゴリズム＋データ構造＝プログラム」というものがあります。

構造化設計ではアルゴリズム重視、オブジェクト指向設計ではデータ構造重視という特徴があります。

詳細設計、すなわちアルゴリズム設計の段階では、使用するプログラミング言語とは独立して自然言語（日本語など）で記述します。つまり、まだPythonやJavaなどの言語は用いません。

この段階では、アルゴリズムの実行手順や必要なデータ構造が明確になります。

ソフトウェア設計は非常に重要であり、保守性や再利用性に大きな影響を与えます。

保守性とは、不具合が見つかった際にその原因を特定し修正しやすいことです。また、ソフトウェアに新たな機能を追加する際の修正容易性も含まれます。

再利用性とは、一度作成したモジュールを他のソフトウェア開発にも流用できることです。再利用が可能であれば、開発コストが下がり、高品質で安価なソフトウェア開発が実現可能になります。

このような効果を得るためには、しっかりとした設計が不可欠です。ソフトウェア設計には必ずしも唯一の「正解」があるわけではなく、複数の設計案の中から、利点と欠点を比較して最適なものを選ぶ必要があります。

つまり、ソフトウェア設計は反復と試行錯誤を伴うプロセスであり、初期案が最適とは限りません。よく考えると別の設計方法の方が効率的であるということも多くあります。

設計でしっかりと工夫すれば、後の保守性、再利用性、処理速度のすべてに良い影響を及ぼすため、非常に重要な工程なのです。

この要件定義と設計を担当する職種が「ITアーキテクト」です。

ITアーキテクトは、顧客が抱えるビジネス上や技術的な課題を分析し、それを解決するための要件を整理・再構成します。その際に技術的リスクを評価し、開発コストに与える影響を見積もることも求められます。

また、ハードウェアやソフトウェアの技術を駆使して、顧客のニーズを満たすシステムのアーキテクチャを設計します。その設計により、情報システム全体の整合性や一貫性を保つ必要があります。

さらに、設計されたアーキテクチャが課題解決に資するものであるか、予定されたコストと期間で導入可能かどうかも確認します。

このような重要な役割を担うITアーキテクトは、現在、IT人材の中でも特に不足しているといわれています。

次に、コーディングの話に移ります。

ここまでで、仕様の決定、モジュールの設計（基本設計）、アルゴリズムやデータ構造の設計（詳細設計）までが完了しています。

コーディングは、これらの設計に基づき、ソフトウェアをプログラミング言語で実装する工程です。

基本設計、詳細設計が済んでいれば、コーディングはある意味で単純作業になります。プログラムの宣言部、つまり入力と出力の定義を行い、アルゴリズムの処理手順をコメントとしてプログラムに埋め込みます。そのコメントに従って実際のコードを記述し、アルゴリズムとコードが整合していることを確認します。

こうしてすべての作業が完了すれば、コーディングは終了となります。

プロジェクトの規模が大きくなると、個人での開発は現実的ではありません。例えば、100億円規模のプロジェクトでは、プログラム行数も数百万行に及び、一人で開発するのは不可能です。

このため、ソフトウェアをいくつかの部分に分割し、それぞれを開発者や開発チームに割り当てます。チームごとに分担して開発し、最後にすべてを統合して全体のソフトウェアを完成させるという手順を取ります。その際、どのようにソフトウェアを分割するかの戦略を立てる必要があります。

ソフトウェアの分割には、大きく分けて二つの方針があります。一つは機能に基づいて分割する構造化設計、もう一つはソフトウェアが管理するデータに基づいて分割するオブジェクト指向設計です。

構造化設計は伝統的に用いられてきましたが、近年では仕様変更に柔軟に対応できるオブジェクト指向設計が主流となっています。どちらを採用するかによって設計の進め方が大きく異なるため、適切な手法の選択が重要です。

構造化設計ではアルゴリズムを中心に、オブジェクト指向設計ではデータ構造を中心にモジュール（クラス）を設計します。各モジュールが提供する機能やルーチン（メソッド）の仕様を定義し、個別の引数や戻り値、必要なデータ構造を決定します。

このように、アルゴリズムとデータ構造は密接に関係しています。実際、「アルゴリズム＋データ構造＝プログラム」という有名な言葉もあります。

詳細設計では、プログラミング言語に依存せず自然言語（例：日本語）でアルゴリズムとデータ構造を記述します。使用する言語（Python、Java、C#など）はこの段階では考慮しません。これにより、各機能の実現手順と必要なデータ構造が明確になります。

ソフトウェア設計は保守性や再利用性に大きく影響します。設計の善し悪しは、後のデバッグのしやすさや機能追加のしやすさに直結します。特に再利用性が高い設計は、他のプロジェクトでもモジュールを使い回せるため、開発コストの削減につながります。

ソフトウェア設計には正解が一つではなく、複数の選択肢の中から利害得失を踏まえて最適なものを選ぶ必要があります。そのため、設計は試行錯誤と反復を伴うプロセスとなります。

設計の質はソフトウェアの性能にも影響します。良い設計を行うことで処理効率の向上も期待できます。したがって、設計段階でしっかり考え抜くことが重要です。

これらの設計や要件定義を担う職種として「ITアーキテクト」があります。ITアーキテクトは、顧客のビジネス課題や技術的課題を分析し、必要な要件を整理・再構成します。また、技術的リスクを評価し、情報システム全体のアーキテクチャを設計します。

この設計には整合性や一貫性を保ち、費用や期間の範囲内での実現可能性を確認することが求められます。ITアーキテクトは現在、IT人材の中でも特に不足している重要な職種の一つとされています。

次にコーディングについて説明します。基本設計、アルゴリズム設計、データ構造の設計が完了した段階で、これに従ってプログラミング言語で実装するのがコーディングです。

この段階では設計が完了しているため、基本的にコーディングは単純作業になります。設計に基づいて宣言部を作成し、アルゴリズムをコメントで埋め込んだ後、対応するコードを記述します。最後にアルゴリズムとコードの整合性を確認して完了です。

プログラムが短ければ設計からコーディングまで一気に進めることも可能ですが、数百行以上になると通用しません。大規模プログラムでは設計と実装を明確に分けることがプロの手法です。

また、妙なテクニックは避け、素直でわかりやすいコーディングを心がけることが重要です。人間だけでなくコンパイラにも理解しやすくなり、最適化が促進されて処理速度が向上します。

続いてソフトウェアテストについて説明します。テストとは、ソフトウェアに入力データを与えて実行結果を得て、それが仕様通りであるかを確認する作業です。期待される出力結果（仕様ベース）と実行結果を照合し、差異があればバグとして特定します。

実際のテストでは、数百から数万、場合によっては億単位のテストケースを用いて検証します。これは、バグが存在する前提で進めるからです。

人間がソフトウェアを作る以上、バグは避けられません。テストを通じて完璧性を証明することはできませんが、バグの数を減らし、運用時のトラブルを抑えることが目的です。

これらの工程を担当する職種として、ソフトウェアデベロップメント、ITスペシャリスト、アプリケーションスペシャリストがあります。

ソフトウェアデベロップメントは、ソフトウェア製品の企画、設計、開発を担います。特に設計と開発を中心に担当し、SEやプログラマーがこれに該当します。

ITスペシャリストは、ハードウェアやネットワーク、セキュリティなどITの専門技術領域に特化した技術者です。一方、アプリケーションスペシャリストは、銀行や航空業など業種固有の業務に精通した専門家です。

実際のソフトウェア開発では、この両方のスペシャリストが協力して業務を遂行する必要があります。

ソフトウェアの出荷後は運用と補修が始まります。運用では、顧客のコンピューターへの導入やユーザー教育、問い合わせ対応などを行います。

補修では、運用中に発見されたバグの修正や仕様変更への対応、新機能の追加、OSの変更への対応などを行います。ソフトウェア開発にかかるコストの40〜80%は、この保守に費やされるとも言われています。

保守作業では、まず既存のプログラムの理解が必要となり、それに基づいてどの部分をどう修正するかを検討する必要があります。設計が複雑だった場合、理解に非常に時間がかかります。

このような運用・補修を担当する職種には、カスタマーサービスとITサービスマネジメントがあります。カスタマーサービスは導入作業や修理対応を行い、ITサービスマネジメントはSLA（サービスレベル合意）に基づいてシステムの安定稼働を支援します。

これでITスキル標準における11職種の説明が一通り終わりました。ここからは、その他の職種として「プロジェクトマネジメント」と「エジュケーション（教育・人材育成）」について触れます。

まず、プロジェクトマネジメントについて説明します。ソフトウェア開発、企画、運用といった各プロセスは、いずれもプロジェクト単位で実施されます。プロジェクトマネージャーの役割は、そうした各プロジェクトの提案、チームの立ち上げ、計画の立案、計画に基づいた実行、予期しない事態への対処、そして最終的なプロジェクトの終結までを管理することです。プロジェクトは一人で進めることはできません。プロジェクトチームとしての成果を出すためには、プロジェクトマネージャーが適切に管理し、メンバーが円滑に働けるよう支援することが不可欠です。このように、プロジェクトマネジメントの職種は非常に重要な位置づけとなっています。

次に、エジュケーションです。これは、直接ソフトウェア開発に関わるのではなく、それに携わる人材の教育を担う職種です。教育対象者に応じて新入社員向け研修、中堅・ベテラン技術者向けのリカレント研修、あるいは経営層向けにビジネス戦略や情報システム戦略の最新動向を解説する研修などがあります。産業界で働くIT技術者を支える教育分野も、進路の一つとして意義ある選択肢です。

以上で、ITスキル標準に基づく11職種に加え、プロジェクトマネジメントとエジュケーションを含めた職種の紹介は終了です。

来週および再来週の授業では、情報系の職種に関する調査を行ってもらいます。各学生に職種を割り当て、i コンピテンシ・ディクショナリー（iCD）に基づいて調査を進めてください。

指示としては、まず講義ホームページから「iCD スキルディクショナリー」の該当部分をダウンロードし、目を通しておいてください。また、第13週のレポート課題について、講義ホームページにて希望テーマの選択を行ってください。テーマの選択は本日午前10時より可能になっています。選べるテーマはセールス、プロジェクトマネジメント、ITスペシャリストなど、8種類です。人数が偏らないよう、均等に割り振れるように設定されています。

さらに、ホームページにある「理想をつくるIT技術者」というPDF資料にも目を通しておいてください。これは次回の授業までに読んでおけば十分です。

以上で本日の授業は終了です。iCDスキルディクショナリーのダウンロード、「理想をつくるIT技術者」の資料閲覧、第13週テーマの選択を忘れずに行ってください。次回の授業に備えて準備を進めてください。

フォームの始まり

フォームの終わり