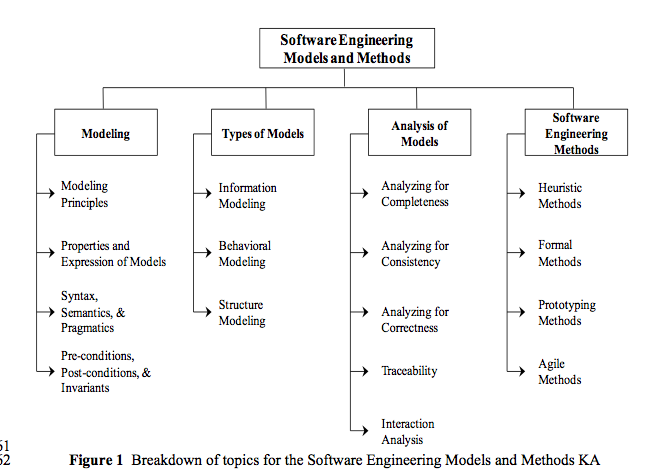
Báo Cáo nghiên cứu

Đề Tài: Đặc tả hình thức prototype trong thiết kế phần mềm

1. **Nội dung đã nghiên cứu**
   1. SOFL: Đọc bài báo SOFL: A formal engineering methodology for industrial application
      1. Qui trình phát triển phần mềm mới được định nghĩa để có thể áp dụng SOFL vào thực tế.
      2. SOFL là sự kết hợp giữa DFD, VDM-SL
      3. Định nghĩa CDFD dựa vào DFD:
         1. Condition process = DeMarco DFD + pre-condition and post-condition
         2. Giới thiệu thêm các kí hiệu Conditional nodes, Connecting points
      4. Định nghĩa các qui tắc để xây dựng mô tả hình thức cho các CDFD bằng việc sử dụng ngôn ngữ VDM
      5. Giới thiệu các qui tắc phân rã mô tả xuống những cấp chi tiết hơn
      6. Giới thiệu khái niệm “”Evolution” cho phép CDFD ngoài việc được phân rã còn có khả năng điều chỉnh để đạt mức độ tốt hơn.
      7. Các nguyên tắc để chuyển đặc tả yêu cầu đã được xây dựng sang thiết kế hướng đối tượng(Object-base design).
      8. Các nguyên tắc để chuyển từ thiết kế sang chương trình
         1. Chuyển trực tiếp: Cây phân cấp CDFD và các module đặc tả kèm theo được chuyển sang cây phân cấp các i-module(mô hình cài đặt).
         2. Chuyển sang mô hình hướng đối tượng: Object-base design sẽ được chuyển thành hệ thống phân cấp của các lớp đối tượng.
      9. Đánh giá về SOFL
         1. CDFD giúp có một cái nhìn toàn cảnh về hệ thống, mô tả hình thức giúp giải quyết những điểm còn mập mờ và giúp cho yêu cầu được đầy đủ và chính xác hơn.
         2. Trong việc trao đổi với khách hàng về việc nắm bắt yêu cầu hệ thống, CDFD giúp khách hàng dễ dàng hiểu được kết quả của việc thu thập yêu cầu về mặc các thành phần của hệ thống(danh sách và sự tương tác giữa chúng) và formal specification giúp khách hàng hiểu chính xác ý nghĩa của từng thành phần được miêu tả bằng CDFD.
         3. SOFL khá linh hoạt trong vấn đề xây dựng đặc tả yêu cầu: đặc tả không nhất thiết phải là 1 hệ thống phân cấp của CDFD hoàn chỉnh mà tuỳ vào người thiết kể có thể thiết kế hệ thống thành nhiều CDFD riêng biệt.
         4. Dễ dàng chuyển đặc tả yêu cầu sang thiết kế và sau đó là thành mô hình cài đặt.
   2. Xem lại DFD  
      Link:  
      <http://en.wikipedia.org/wiki/Data_flow_diagram>
   3. Tìm hiểu bức tranh chung về các mô hình và phương pháp phát triển phần mềm ﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽p phát tri mềm  
      Link : <http://schlumberger.wiki.hci.edu.sg/file/view/Software+Engineering+Models+and+Methods.pdf>  
      Tài liệu tóm lược hầu hết các phương pháp và mô hình xây dựng phần mềm đang được sử bản:﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽mô hình ddụng  
      
      1. Types of models:
         1. Information modeling: Xác định và định nghĩa các khái niệm, thuộc tính, quan hệ và ràng buộc của các thành phần dữ liệu trong hệ thống một cách trừu tượng. Mô hình này được dùng để phân rã thành các mô hình cơ sở dữ liệu của phần mềm.
         2. Behavioral Modeling: Xác định và định nghãi chức năng của hệ thống. Gồm các loại cơ bản:
            1. State machine: mô hình hoá phần mềm bằng các state, event và transition. Phần mềm chuyển từ trạng thái này và trạng thái kia bằng cách thực hiện một loạt các event.
            2. Control-flow model: hệ thống bao gồm các process và được active hoặc deactive bằng một chuỗi các sự kiện.
            3. Data-flow model: Mô tả luồng di chuyển của data qua các process từ data store.
         3. Structure modeling: Mô tả hệ thống dưới dạng cấu trúc. Các lược đồ của UML cung cấp các công cụ để mô hình hoá: class, component, object... diagrams.
      2. Software engineering methods:
         1. Heuristic methods: experience base software engineering
         2. Formal methods: sử dụng các kí hiệu và ngôn ngữ toán học(giảm thiểu sự mập mờ) để mô tả
            1. Specification languages: dựa vào toán học, ngôn ngữ hình thức. Sử dụng trong giai đoạn mô tả hệ thống, phân tích yêu cầu hoặc/và giai đoạn thiết kế.
            2. Program refinement and Derivation:  
               Refinement là quá trình tạo ra các mô tả ở mức thấp hơn từ mô tả ở cao sử dụng các phương pháp chuyển đổi.
            3. Formal verification: Model checking
            4. Logical inference: Mô tả pre-condition, post-condition của các thành phần thiết kế và sử dụng toán logic để chứng minh
         3. Prototyping methods: Tạo ra các phiên bản nhỏ và chưa đầy đủ của hệ thống, được sử dụng để xác định các chức năng mới, trao đổi với khách hàng để nhận phản hồi về việc nắm bắt yêu cầu, thiết kế và cài đặt. Phương pháp này khác với các phương pháp khác ở chỗ nó được sử dụng để hiểu rõ hơn về các khía cạnh chưa rõ ràng của hệ thống(các phương pháp khác bắt đầu với phần được hiểu rõ nhất).
            1. Prototyping style: throwaway, evolution hoặc executable specification.
            2. Prototyping evaluation techniques: prototype có thể được đánh giá và kiểm trả so với việc cài đặt thực tế của phần mềm, hoặc với yêu cầu của hệ thống. Prototype còn được sử dụng như là một mô hình cho việc phát triển hệ thống trong tương lai(user interface )specification.
         4. Agile methos:
            1. Pair programming
            2. RAD
            3. Extreme programming(XP)
            4. Scrum
            5. FDD
   4. Tìm hiểu về “Web service composition as AI planning”  
      *Bài báo này em đã đọc qua nhưng thực sự chưa hiểu thấu đáo do có khá nhiều khái niệm(lĩnh vực) toán học em chưa được biết. Ví dụ: Các chiến lược planning cơ bản: dùng đồ thị, partial order refinement … nên em chỉ có thể tổng hợp một cách sơ lược về cấu trúc của bài báo. Điều này sẽ giúp ích cho việc tìm hiểu kĩ hơn sau này.*  
      Bài báo là một tổng hợp về các phương pháp được sử dụng để xây dựng một agent có khả năng liên kết các webservice trên internet để thực hiện một yêu cầu chức năng nào đó, ví dụ: Với yêu cầu về đi du lịch, agent sẽ tìm kiếm và liên kết tự động các service như: đặt vé máy bay, đặt phòng khách sạn… phù hợp với thông tin của khách hang.  
      Vấn đề đặt ra gồm có 2 thành phần:  
      - Cơ chế để quản lí danh sách các service hiện có. Cơ chế này phải cung cấp một sự tiếp cận dễ dàng cho việc quản lí thay đổi cho các service: dùng ngôn ngữ hình thức để mô tả service.  
      - Phương pháp kết hợp các webservice lại với nhau một cách hiệu quả: AI planning
      1. Web service domains:Shopping, document handling, mail, traveling
      2. Formalizing the planning domain: Hình thức hoá planning domain -> domain theories . Lựa chọn các phương pháp để mô tả hình thức các operation được thực hiện bởi planning agent, thường dựa theo các mô hình chuyển trạng thái(state-transition model). Có 2 loại domain theories:  
          - Dựa vào logic cổ điển  
          - Dựa vào logic mở rộng.
      3. Formalizing the initial world: initial world là trạng những trạng thái ban đầu mà nếu planning agent thực hiện planning trên nó sẽ đạt được mục đích mong muốn. Phương pháp mô tả được dùng chung với planning domain.
      4. Formalizing Goals: desire world state  
         initialize world –-a plan--🡪 desire world state(Goal)  
         Goal specification cũng phải thoả mãn các yêu cầu sau:  
          - Yêu cầu về cấu trúc thời gian: Planning goal cần được chia thành các giai đoạn riêng biệt và liên tiếp, để tránh các trường hợp như: Lập kế hoạch cho 1 chuyến đi từ Vienna -> zurich -> Vienna, goal specification không thể đơn thuẩn có dạng đơn giản “đích đến ”Vienna” vì khi đó Goal state sẽ trùng với initialize state.   
          - Chiến lược để đối phó với trường hợp không xác định được plan  
          - Các thuộc tính an toàn: ràng buộc các điều kiện lên các thuộc tính.  
          - Phân biệt giữa information goal và achievement goal: Ví dụ Goal là lấy thông tin về màu sắc của một đối tượng sẽ không được plan bằng các operation có tác động thay đổi màu sắc của đối tượng đó.  
          - Yêu cầu của người dùng lên plan đưa ra.
      5. Biểu diễn plan: Các biểu diễn truyền thống: plan được biểu diễn như là một chuỗi các operation. Nhưng kiểu biểu diễn đơn giản này không thể đủ để biểu diễn các plan phức tạp. Có một số mở rộng của kiểu cổ điển đã được giới thiệu(ở các phần sau).
      6. Các chiến lược planning cở bản:
         1. Planning dựa vào không gian trạng thái(state-space)
         2. Dựa vào đồ thị
         3. Partial order refinement planning
         4. Planning as satisfiability
            1. Planning as propositional satisfiability
            2. Planning as Description Logic Satisfiability
            3. Planning as Petri-Net Reachability
         5. Planning as Logic Programming
      7. Planning với Control Knowledge
         1. Hierarchical Task Network Planning
         2. High-level Program Execution
         3. Planning As Model Checking
         4. Temporal Planning
2. **Kế hoạch kế tiếp**:
   1. Tìm hiểu sâu hơn về SOLF(mượn sách của thầy), DeMarco DFD
   2. Tìm hiểu sâu hơn và VDM
   3. Tìm hiểu them về software prototype:
      1. Các loại prototype.
      2. Các phương pháp tạo ra prototype.
   4. Tiếp tục tìm hiểu về AI planning