TRƯỜNG ĐẠI HỌC VINH
Viện Kỹ thuật và Công nghệ

PHÂN TÍCH VÀ THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

Th.S. Võ Đức Quang

Bộ môn KHMT&CNPM, Viện Kỹ thuật và công nghệ

✉ quangvd.cntt.dhv@gmail.com, quangvd@vinhuni.edu.vn

☎ 0989.891.418

Chương 3. Mô hình hóa hệ thống

- 3.1. Một số khái niệm mô hình hóa
- 3.2. Ngôn ngữ mô hình hóa thống nhất UML
- 3.3. Các biểu đồ trong UML
  - Các biểu đồ về cấu trúc
  - Các biểu đồ về hành vi

Mô hình hóa

- **Mô hình(model):** là một dạng trừu tượng hóa/một hình ảnh/một biểu diễn của một hệ thống thực, được diễn tả:
  - ở một mức độ trừu tượng hóa nào đó,
  - theo một quan điểm/góc nhìn nào đó,
  - bởi một hình thức diễn tả hiểu được nào đó (như văn bản, đồ thị, phương trình,…)
- **Mô hình hóa: (modeling)** dùng mô hình để nhận thức và diễn tả một hệ thống

✓Quá trình phân tích và thiết kế HT cũng được gọi là quá trình mô hình hóa HT

Mô hình hóa

- Mục đích của mô hình hóa:
  - Để hiểu
  - Để trao đổi
  - Để hoàn chỉnh
- Mô hình hóa tốt phải thỏa các yêu cầu sau:
  - dễ đọc
  - dễ hiểu
  - dễ trao đổi
  - xác thực
  - chặt chẽ
  - đầy đủ
  - dễ thực hiện (cài đặt)

Mô hình hóa

- Kết hợp 3 thành phần:
  - Hệ ký pháp (notation): Các khái niệm và mô hình
  - Một tiến trình (process): Các bước cần tiến hành, các sản phẩm (tài liệu, mô hình) qua từng giai đoạn, cách điều hành tiến trình, cách đánh giá chất lượng
- Công cụ hỗ trợ: Phần mềm hỗ trợ cho quá trình mô hình hóa. Yêu cầu có khả năng:
  - Sản sinh các mô hình và biểu đồ,
  - Biến đổi và điều chỉnh nhanh các mô hình và biểu đồ,
  - Kiểm tra cú pháp, sự chặt chẽ, đầy đủ,
  - Kiểm thử và đánh giá,
  - Mô phỏng thực hiện mô hình

Các mô hình lập trình

- Mô hình lập trình thủ tục: Pascal, C, Ada, Cobol,…
  - Mô hình lập trình logic: Prolog, C5,…
  - Mô hình lập trình hàm: Matlab, Lisp, Haskell,…
  - Mô hình lập trình hướng đối tượng: C++, Java,…

Quy trình RUP

- RUP (Rational Unified Process) là một quy trình mô hình hóa với UML, không phải là một chuẩn
  - Các nguyên tắc cơ bản
  - Các giai đoạn chính (phases)
  - Các bước chính (steps)

Quy trình RUP

- Lặp và tăng trưởng
  - Dự án được chia thành những vòng lặp hoặc giai đoạn ngắn để dễ dàng kiểm soát
  - Cuối mỗi vòng lặp, phần thi hành được của hệ thống phần mềm được sản sinh theo cách thêm vào dần dần
- Tập trung vào kiến trúc
  - Hệ thống phức tạp được chia thành các mô-đun để dễ dàng triển khai và bảo trì
  - Kiến trúc này được trình bày theo **05 góc nhìn khác nhau**

Quy trình RUP

- Dẫn dắt theo các ca sử dụng (use cases)
  - Nhu cầu người dùng thể hiện bởi các ca sử dụng. Các ca sử dụng ảnh hưởng xuyên suốt cho mọi giai đoạn phát triển hệ thống, là cơ sở xác định vòng lặp và tăng trưởng, là căn cứ để phân chia công việc trong nhóm
  - Nắm bắt nhu cầu: Phát hiện các ca sử dụng
  - Phân tích: Đi sâu vào mô tả các ca sử dụng
  - Thiết kế và cài đặt: Xây dựng hệ thống theo các ca sử dụng
  - Kiểm thử và nghiệm thu hệ thống: Thực hiện theo các ca sử dụng
  - Khống chế các nguy cơ (risks)
  - Phát hiện sớm và loại bỏ các nguy cơ đối với dự án PTPM

Quy trình RUP

- RUP được tổ chức thành **04 giai đoạn**:
  - Khởi đầu (Inception),
  - Chi tiết hóa (Elaboration),
  - Xây dựng (Construction), và
  - Chuyển giao (Transition)

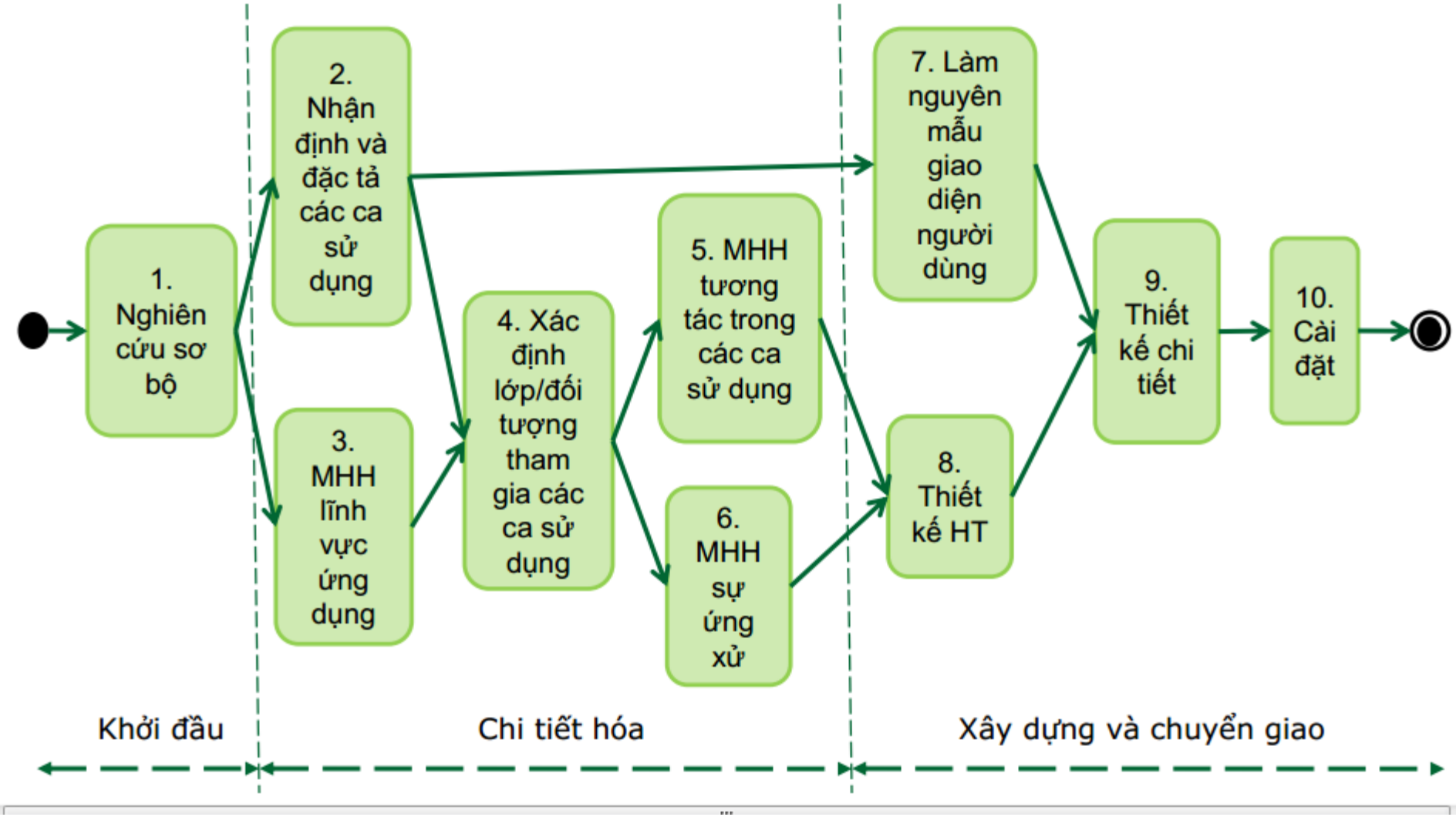
Quy trình RUP

- Giai đoạn khởi đầu (Inception)
  - Cho một cái nhìn tổng quát về hệ thống phần mềm (chức năng, hiệu năng, công nghệ,…) và về dự án PTPM sẽ triển khai (phạm vi, mục tiêu, tính khả thi,…) => Đưa ra kết luận nên phát triển dự án hay loại bỏ?
- Giai đoạn chi tiết hóa (Elaboration)
  - Phân tích chi tiết hơn về hệ thống (chức năng và cấu trúc tĩnh)
  - Đề xuất một kiến trúc hệ thống (nguyên mẫu)

Quy trình RUP

- Giai đoạn xây dựng (Construction)
  - Tập trung vào thiết kế và cài đặt hệ thống
  - Kiến trúc hệ thống được chi tiết hóa, chỉnh sửa
  - Kết thúc khi đã cho ra được 1 hệ thống hoàn chỉnh với tài liệu kỹ thuật đi kèm
  - Là giai đoạn tốn nhiều thời gian, công sức nhất
- Giai đoạn chuyển giao (Transition)
  - Chuyển giao hệ thống cho người dùng cuối: chuyển đổi dữ liệu, lắp đặt, kiểm tra, đào tạo,…

Quy trình RUP – Các bước chính



Quy trình RUP

1. Nghiên cứu sơ bộ
   - Đưa ra cái nhìn khái quát về hệ thống phần mềm và dự án PTPM
   - Đưa ra kết luận: nên/không nên triển khai dự án?
2. Nhận định và đặc tả các ca sử dụng
   - Nắm bắt nhu cầu của người dùng, phát hiện các ca sử dụng
   - Mỗi ca sử dụng phải được đặc tả (được mô tả) dưới dạng kịch bản và/hoặc một biểu đồ trình tự
3. MHH lĩnh vực ứng dụng
   - Đưa ra mô hình biểu đồ lớp phản ánh mọi khái niệm, nghiệp vụ
   - Các lớp ở đây là các lớp lĩnh vực (không phải là các lớp đối tượng)

Quy trình RUP

4. Xác định các đối tượng/lớp tham gia ca sử dụng
   - Với mỗi ca sử dụng, phát hiện các lớp lĩnh vực, lớp điều khiển, lớp biên
5. MHH tương tác trong các ca sử dụng
   - Các đối tượng tương tác bằng cách trao đổi thông điệp
   - Tạo kịch bản của ca sử dụng: biểu đồ trình tự, biểu đồ giao tiếp
6. MHH sự ứng xử
   - Các đối tượng điều khiển có khả năng ứng xử đối với các sự kiện đến từ bên ngoài để điều khiển
   - Sử dụng biểu đồ trạng thái để mô tả hành vi ứng xử của các đối tượng điều khiển

Quy trình RUP

7. Làm nguyên mẫu giao diện người dùng
   - Sử dụng các bộ tạo lập giao diện người dùng (graphical user interface – GUI) để làm sớm nguyên mẫu giao diện, giúp cho việc mô hình hóa và cài đặt hệ thống dễ dàng hơn
8. Thiết kế hệ thống
   - Thiết kế kiến trúc tổng thể của hệ thống
   - Chia thành các hệ thống con, chọn lựa loại hình điều khiển thích hợp
   - Dùng biểu đồ thành phần mô tả các thành phần vật lý
   - Dùng biểu đồ triển khai mô tả cách bố trí, triển khai các thành phần thực thi của hệ thống vào các phần cứng và nền tảng hạ tầng
   - Kiến trúc khách/chủ (client/server) là một kiến trúc hệ thống hay được sử dụng

Quy trình RUP

9. Thiết kế chi tiết
   - Thiết kế các lớp, các liên kết, các thuộc tính, các phương thức
   - Xác định các giải pháp cài đặt hệ thống
10. Cài đặt
   - Lập trình và kiểm thử
   - Hệ thống được nghiệm thu dựa theo các ca sử dụng

Quy trình RUP – Công cụ hỗ trợ

- Hỗ trợ việc lập trình phát triển hệ thống (Integrated Development Environment – IDE)
  - Soạn thảo, biên dịch
  - Gỡ lỗi, kiểm thử
  - Xây dựng nguyên mẫu giao diện
- Hỗ trợ việc mô hình hóa (Modeling tools)
  - Sản sinh, biến đổi, điều chỉnh các mô hình và biểu đồ
  - Kiểm tra cú pháp của các mô hình
  - Lưu trữ và quản lý phiên bản các mô hình
  - Kiểm thử và đánh giá các mô hình
  - Mô phỏng và thực hiện mô hình
  - Sinh ngược mô hình từ phần mềm có sẵn

Quy trình RUP – Công cụ hỗ trợ

- Hỗ trợ quy trình phát triển hệ thống
  - Dẫn dắt và hỗ trợ trực tuyến, chỉ rõ công việc và sản phẩm của các giai đoạn
  - Hỗ trợ tiến trình lặp
  - Hỗ trợ làm việc nhóm
  - Tích hợp được với các công cụ (tools) khác
  - Trợ giúp quản lý, lên kế hoạch, theo dõi quá trình thực hiện dự án

UML

- Lịch sử phát triển của ngôn ngữ mô hình hóa UML
- Các góc nhìn của UML
- Các biểu đồ được sử dụng trong UML
- Các công cụ mô hình hóa UML miễn phí

UML – Lịch sử phát triển

- Ngôn ngữ UML (Unified Modeling Language) là một hệ thống ký pháp mô hình hóa hướng đối tượng
- 1975-1990:
  - Có nhiều ngôn ngữ MHH hướng đối tượng được phát triển
- 1990-1994:
  - Hơn 50 phương pháp phát triển hướng đối tượng, trong đó có 3 phương pháp nổi tiếng:
    - OOD - Object Oriented Design (Grady Booch)
    - OOSE - Object Oriented Software Engineering (Ivar Jacobson)
    - OMT - Object Modeling Technique (Jim Rumbaugh)

UML – Lịch sử phát triển

- 10/1994: Rumbaugh và Booch tiến hành dự án UML ở Rational, xây dựng một phương pháp hợp nhất trên cơ sở hai phương pháp Booch 93 và OMT-2
- 1995: Jacobson gia nhập dự án
- 10/1995: Phác thảo UML, phiên bản 0
- 6/1996: Phiên bản UML 0.9
- 1/1997: IBM và SoftTeam kết hợp với các thành viên => Phiên bản 1.1
- 14/11/1997: UML 1.1 được OMG (Object Management Group) công nhận là chuẩn
- 6/1998: UML 1.2
- 10/1998: UML 1.3
- 5/2001: UML 1.4
- 6/2003: UML 2.0

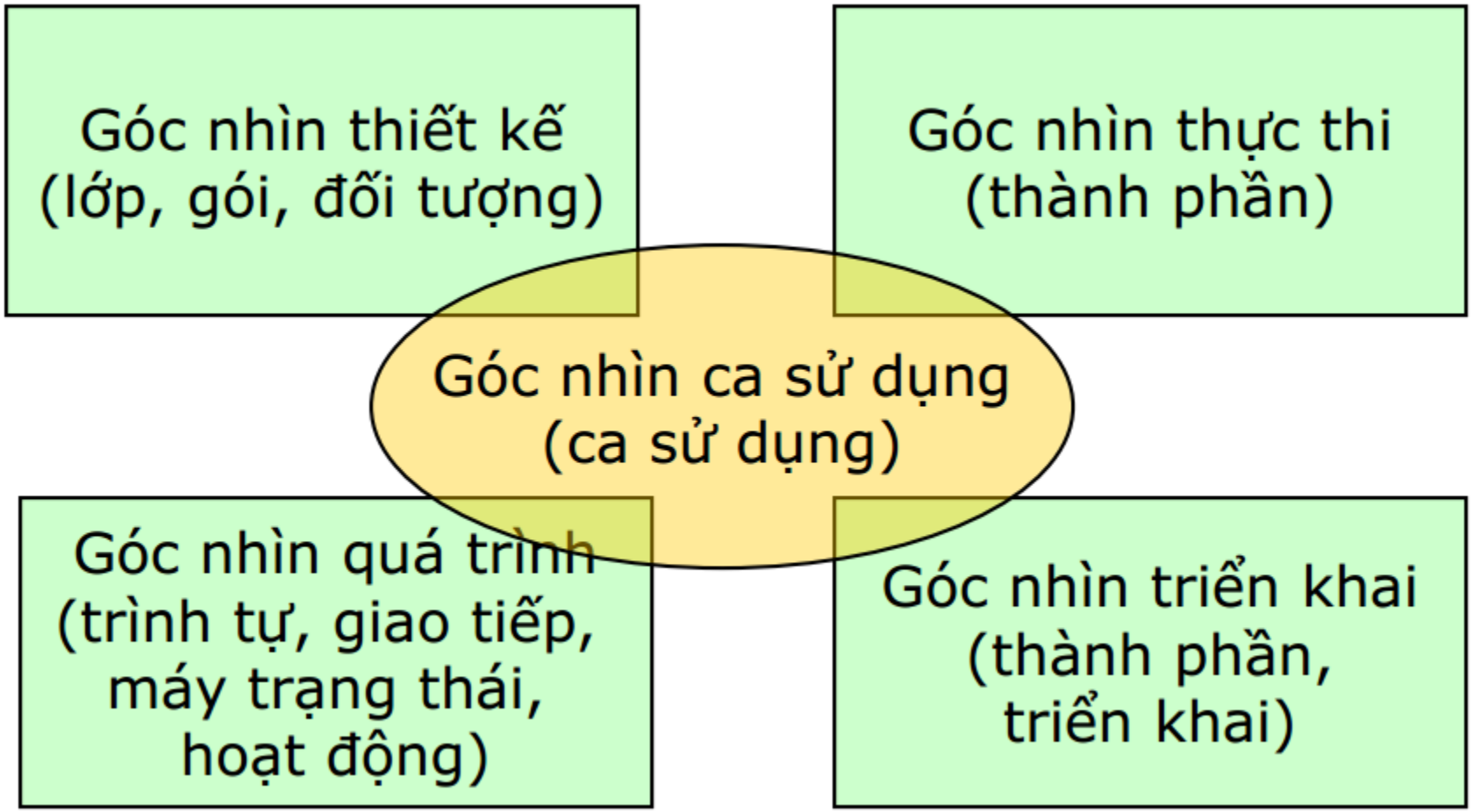
UML – Sử dụng

- UML là ngôn ngữ dùng để:
  - Mô hình hóa trực quan (Visualizing)
  - Đặc tả (Specifying)
  - Xây dựng (Constructing)
  - Làm tài liệu (Documenting)
- Có thể sử dụng trong bất kỳ tiến trình phát triển hệ thống
- Xuyên suốt vòng đời phát triển hệ thống
- Được sử dụng bởi các công nghệ cài đặt khác nhau

UML – Các góc nhìn

- UML cung cấp các mô hình để diễn tả hệ thống
- Mỗi mô hình chỉ có thể diễn tả hệ thống theo một góc nhìn (view) nhất định
- UML cung cấp 05 góc nhìn đối với hệ thống
- Mỗi góc nhìn thực hiện bởi một số biểu đồ (mô hình)
- Có thể có biểu đồ thuộc vào nhiều (>1) góc nhìn khác nhau

UML – Các góc nhìn



UML – Các góc nhìn

- **Góc nhìn ca sử dụng (Use case view)**
  - Là góc nhìn từ ngoài nhìn vào hệ thống
  - Là cách nhìn của người dùng cuối, người phân tích, người kiểm thử
  - Không phản ánh tổ chức bên trong, mà chỉ làm rõ các chức năng chính/quan trọng mà hệ thống phải đáp ứng cho người dùng
  - Sắc thái tĩnh: Biểu đồ ca sử dụng (Use case diagram)
  - Sắc thái động: Biểu đồ giao tiếp (Communication diagram), Biểu đồ máy trạng thái (State diagram), và Biểu đồ hoạt động (Activity diagram)

UML – Các góc nhìn

- **Góc nhìn thiết kế (Design view)**
  - Còn được gọi là góc nhìn logic (Logical view)
  - Là góc nhìn vào bên trong (cấu trúc) hệ thống, cho thấy các nhiệm vụ của hệ thống
  - Là cách nhìn của người thiết kế hệ thống
  - Sắc thái tĩnh: Biểu đồ lớp (Class diagram), Biểu đồ đối tượng (Object diagram)
  - Sắc thái động: Biểu đồ giao tiếp (Communication diagram), Biểu đồ máy trạng thái (State diagram), Biểu đồ hoạt động (Activity diagram)

UML – Các góc nhìn

- **Góc nhìn quá trình (Process view)**
  - Còn được gọi là góc nhìn song hành
  - Phản ánh các quá trình điều khiển, các quá trình thực hiện, cho thấy sự hoạt động đồng bộ của hệ thống
  - Được thể hiện (sử dụng) với các biểu đồ như trong Góc nhìn thiết kế, tập trung vào các lớp chủ động
    - Lớp chủ động: Lớp biểu diễn cho các quá trình điều khiển và quá trình thực hiện

UML – Các góc nhìn

- **Góc nhìn thực thi (Implementation view)**
  - Còn được gọi là góc nhìn thành phần (Component view)
  - Là góc nhìn đối với dạng phát hành của phần mềm
  - Cho thấy các thành phần và tập tin tương đối độc lập, có thể lắp ráp để hệ thống chạy được
  - Sắc thái tĩnh: Biểu đồ thành phần (Component diagram)
  - Sắc thái động: Biểu đồ giao tiếp (Communication diagram), Biểu đồ máy trạng thái (State diagram),
  - Biểu đồ hoạt động (Activity diagram)

UML – Các góc nhìn

- **Góc nhìn triển khai (Deployment view)**
  - Là góc nhìn về kiến trúc phần cứng và nền tảng hạ tầng mà trên đó hệ thống được triển khai
  - Chỉ rõ sự phân bố, sắp đặt các thành phần của hệ thống trên các đơn vị phần cứng và nền tảng hạ tầng
  - Sắc thái tĩnh: Biểu đồ triển khai (Deployment diagram)
  - Sắc thái động: Biểu đồ giao tiếp (Communication diagram), Biểu đồ máy trạng thái (State diagram),
  - Biểu đồ hoạt động (Activity diagram)

UML – Các góc nhìn

- Mỗi vai trò trong tiến trình phát triển hệ thống (vd: phân tích, thiết kế, tích hợp, kiểm định, người dùng cuối,…) thường chỉ quan tâm tới một góc nhìn nào đó của hệ thống
- 05 góc nhìn có sự liên hệ và bổ trợ lẫn nhau
- Góc nhìn ca sử dụng (Use case view) có ảnh hưởng (liên quan) đến 04 góc nhìn còn lại

UML – Các biểu đồ sử dụng

- Các biểu đồ về cấu trúc:
  - ✓**Biểu đồ lớp (Class diagram)**
  - ✓Biểu đồ đối tượng (Object diagram)
  - ✓**Biểu đồ triển khai (Deployment diagram)**
  - ✓Biểu đồ gói (Package diagram)
  - ✓**Biểu đồ thành phần (Component diagram)**
  - ✓Biểu đồ cấu trúc đa hợp (Composite structure diagram)
- Các biểu đồ về hành vi:
  - ✓**Biểu đồ ca sử dụng (Use case diagram)**
  - ✓**Biểu đồ hoạt động (Activity diagram)**
  - ✓**Biểu đồ trình tự (Sequence diagram)**
  - ✓Biểu đồ giao tiếp (Communication diagram),
  - ✓**Biểu đồ máy trạng thái (State diagram)**
  - ✓Biểu đồ thời gian (Timing diagram)
  - ✓Biểu đồ tổng quan tương tác (Interaction overview diagram)

UML – Các phần tử của biểu đồ

- Các nút (node)
  - Các yếu tố của mô hình (biểu đồ)
  - Có dạng đồ họa 2 chiều
  - Vd: Lớp, Gói,…
- Các đường (path)
  - Các yếu tố của mô hình (biểu đồ)
  - Có dạng đồ họa tuyến tính
  - Vd: Liên kết, Phụ thuộc, Khái quát

UML – Một số định nghĩa

- Đặc tả (specification)
  - Một phát biểu dạng văn bản (textual statement) về cú pháp và ngữ nghĩa
- Tô điểm (adornment)
  - Các vai trò, các cơ số, hạn định (giới hạn), đường viền đậm nét,…
- Khuôn dập (stereotype)
  - Chuỗi ký tự, được đặt trong ngoặc kép, biểu tượng gắn thêm vào để tạo phần tử mới
- Tính chất (property) và giá trị gán nhãn (tagged value)
  - Đưa thêm thông tin cho các phần tử mô hình
  - Vd: {nhãn=giá trị}, {nhãn_boolean}
- Ràng buộc (constraint)
  - Thêm các điều kiện/hạn chế đối với một yếu tố của mô hình

Mô hình hóa với UML

- Mô hình hóa hệ thống theo nhiều góc nhìn
  - Có thể sử dụng 05 góc nhìn đối với hệ thống
  - Tùy vào hệ thống nhỏ/lớn, đơn giản/phức tạp => Quyết định mô tả hệ thống theo những góc nhìn phù hợp
- Mô hình hóa hệ thống theo nhiều mức độ trừu tượng hóa khác nhau (different abstraction levels)
  - Tùy thuộc vào giai đoạn (của tiến trình phát triển hệ thống) và nhu cầu sử dụng
  - Có thể ở mức khái quát (logical/overview level) hoặc ở mức chi tiết (detailed level)

UML

- ArgoUML (http://argouml.tigris.org/)
  - Từ mô hình sinh mã nguồn (diagram-to-codes generation) cho các ngôn ngữ: C++, C#, Java, PHP4, PHP5, Ruby
  - Từ mã nguồn sinh ngược lại mô hình (reverse engineering) cho các ngôn ngữ: Java
- BOUML (http://www.bouml.fr/)
  - Từ mô hình sinh mã nguồn cho các ngôn ngữ: C++, Java, PHP, IDL, Python, MySQL
  - Từ mã nguồn sinh ngược lại mô hình cho các ngôn ngữ: C++, Java, PHP, MySQL
- NClass (http://nclass.sourceforge.net/)
  - Từ mô hình sinh mã nguồn cho các ngôn ngữ: C#, Java
  - Từ mã nguồn sinh ngược lại mô hình cho các ngôn ngữ: C#, Java
- Umbrello UML Modeller (https://umbrello.kde.org/)
  - Từ mô hình sinh mã nguồn cho các ngôn ngữ: C++, Java, Perl, PHP, Python
  - Từ mã nguồn sinh ngược lại mô hình cho các ngôn ngữ: C++, IDL, Pascal/Delphi, Ada, Python, Java

UML

- WhiteStarUML (https://sourceforge.net/projects/whitestaruml/)
  - Từ mô hình sinh mã nguồn cho các ngôn ngữ: Java, C#, C++, SQL
  - Từ mã nguồn sinh ngược lại mô hình cho các ngôn ngữ: Java, C#, C++, SQL
- Open ModelSphere (http://www.modelsphere.com/org/)
  - Từ mô hình sinh mã nguồn cho các ngôn ngữ: Java, SQL
  - Từ mã nguồn sinh ngược lại mô hình cho các ngôn ngữ: Java
- Modelio (https://www.modelio.org/)
  - Từ mô hình sinh mã nguồn cho các ngôn ngữ: Java
  - Từ mã nguồn sinh ngược lại mô hình cho các ngôn ngữ: Java
- Dia (https://wiki.gnome.org/Apps/Dia/)
  - Từ mô hình sinh mã nguồn cho các ngôn ngữ: Python, C++, JavaScript, Pascal, Java, PHP
- Papyrus (http://www.eclipse.org/papyrus/)
  - Từ mô hình sinh mã nguồn cho các ngôn ngữ: C/C++, Java
- Acceleo (https://www.eclipse.org/acceleo/)
  - Từ mô hình sinh mã nguồn cho các ngôn ngữ: JEE, .Net, Php, ...

UML – Yêu cầu về nhà

- Mỗi SV cài đặt StarUML, đọc trước các hướng dẫn để vẽ
  - Biểu đồ ca sử dụng (Use case diagram)
  - Biểu đồ hoạt động (Activity diagram)
  - Biểu đồ trình tự (Sequence diagram)
- Nhóm **[?][?][?][?]** sử dụng biểu đồ Use case để mô hình hóa các ca sử dụng máy rút tiền tự động ATM.
- Nhóm **[?][?][?][?]** sử dụng biểu đồ Use case để mô hình hóa các ca sử dụng hệ thống mua hàng online.