Faculty of Arts and Science
University of Toronto

Midterm Test

Department: Computer Science
Instructor: Matt Medland
Date & Time: 10:10 am - Tuesday, Feb 26, 2013

Conditions: Closed Book
Duration: 50 minutes

This test counts for 15% of your final grade

Name: _____
(Please underline last name)

Student Number: _____

Question Marks

1 _____ /25
2 _____ /25
3 _____ /10
4 _____ /10

Total _____ /70 = _____ %

1. [Modeling Questions, 25 marks Total]

(a) [Why Build Models? – 5 marks] As software developers, we often build models during design and/or we reverse-engineer them from an existing system. List five (5) reasons for building these models.

One mark for each, up to five (5). Some examples:

- Abstraction: ignore some details to get the big picture
- Decomposition: break a problem into manageable chunks
- Modularization: break into packages/modules of related functionality
- Communication: discuss design alternatives with others
- Discovery: help to figure out what questions to ask
- Problem identification: find problems earlier
- Other reasonable answers acceptable too...

(b) [UML Diagram Types – 5 marks] Briefly describe what each of the following types of UML diagrams are typically used for:

- Use case diagram

Show interactions between roles/actors and the system.

- Class diagram

Depicts a system's classes, attributes, operations and relationships

- Sequence diagram

An interaction diagram showing how classes operate with each other and in what order.

- Package Diagram

Depicts dependences between packages/modules in a given model.

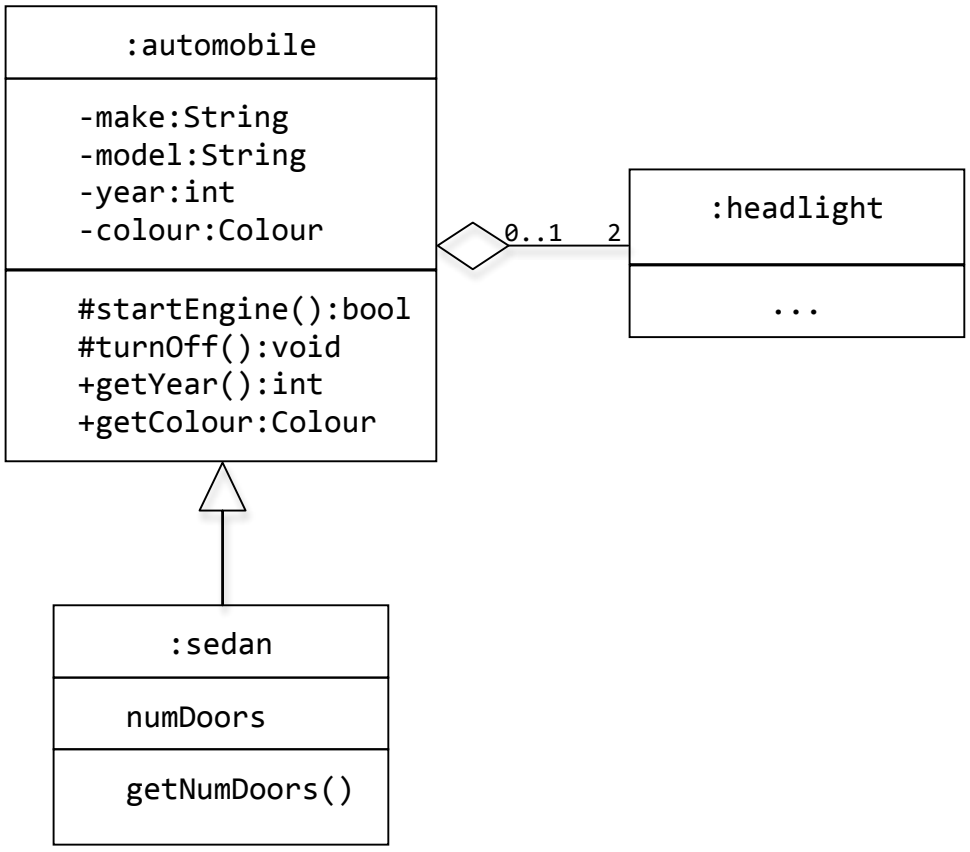
- Robustness diagram (not really UML)

Helps to bridge the gap between requirements & design. Generally between use case & sequence diagram. Throw them away when done.

(c) [UML Class Diagram Elements – 10 marks] Identify as many of the items from the list in the UML class diagram. Marks will be awarded for correct identification, and subtracted for incorrect answers. Not all elements in the list appear in the diagram.

Class Diagram Elements

- generalization
- object return type
- private attribute
- private operation
- realization
- protected operation
- composition
- aggregation
- exception
- class name
- stereotype
- dependency
- public attribute
- public operation
- multiplicity
- visibility modifier
- primitive attribute



(d) [Multiplicity – 5 marks] Relationships in UML diagrams often have multiplicities associated with them. Briefly describe, in English, what each of the multiplicities below mean. Marks are awarded for correct answers and subtracted for incorrect answers. Maximum of five (5) marks.

- 0..1

Zero or one

- 1

Exactly one

- n (Natural number)

Exactly n

- 0..n

Between zero and n, inclusive

- 1..n

Between one and n, inclusive

- *

Zero or more

- 0..*

Zero or more

- 1..*

One or more

2. [UML Diagrams, 25 marks Total]
(a) [Design Patterns & Sequence Diagrams – 15 marks] Recall from the lectures that we created a UML sequence diagram from the Observer pattern. See Figures 1 & 2 below for a refresher.

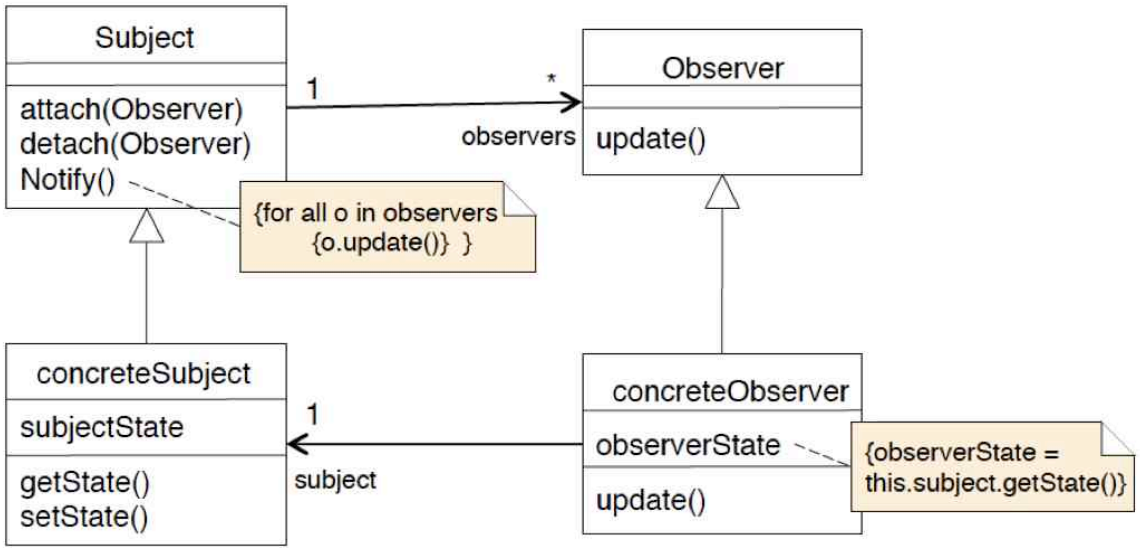


Figure 1: Observer Pattern Class Diagram

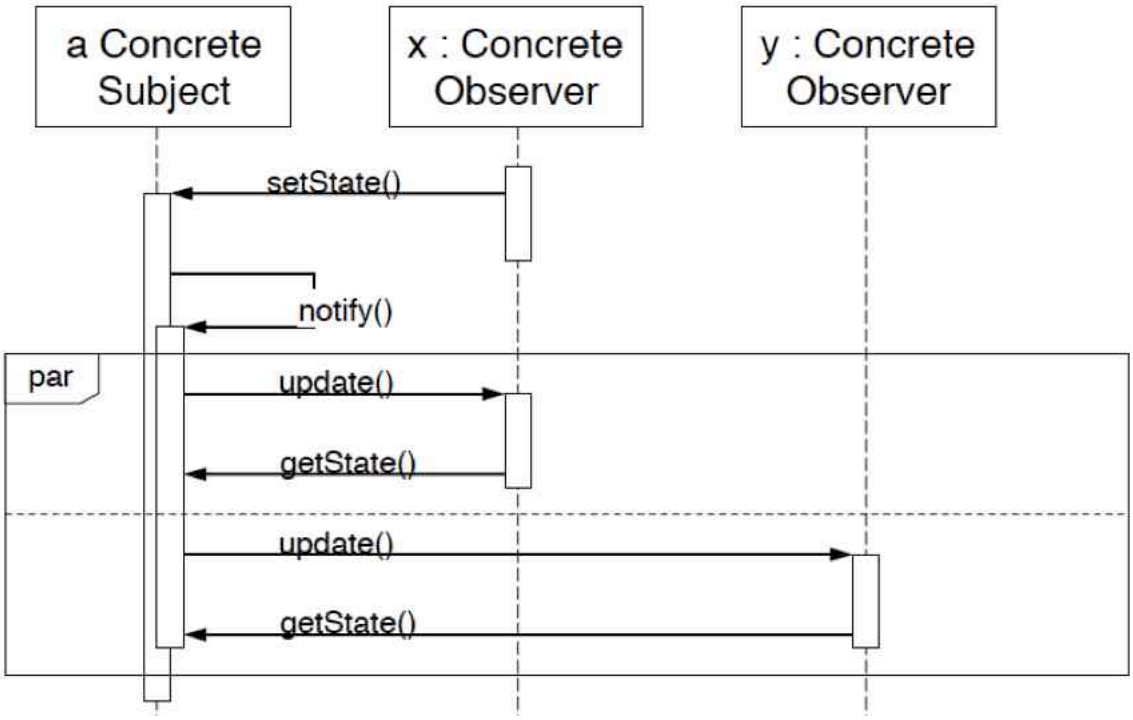


Figure 2: Observer Pattern Sequence Diagram

For this question, you are to draw a sequence diagram for the Iterator pattern. The class diagram for Iterator is shown in Figure 3 below. Use page 7 (and the scratch paper at the back of this package, if necessary) to draw a UML Sequence diagram representing the Iterator pattern.

Hints:

- you do not need abstract classes or interfaces in your diagram
- you will need at least one interaction frame
- there is one message to which you can apply a stereotype

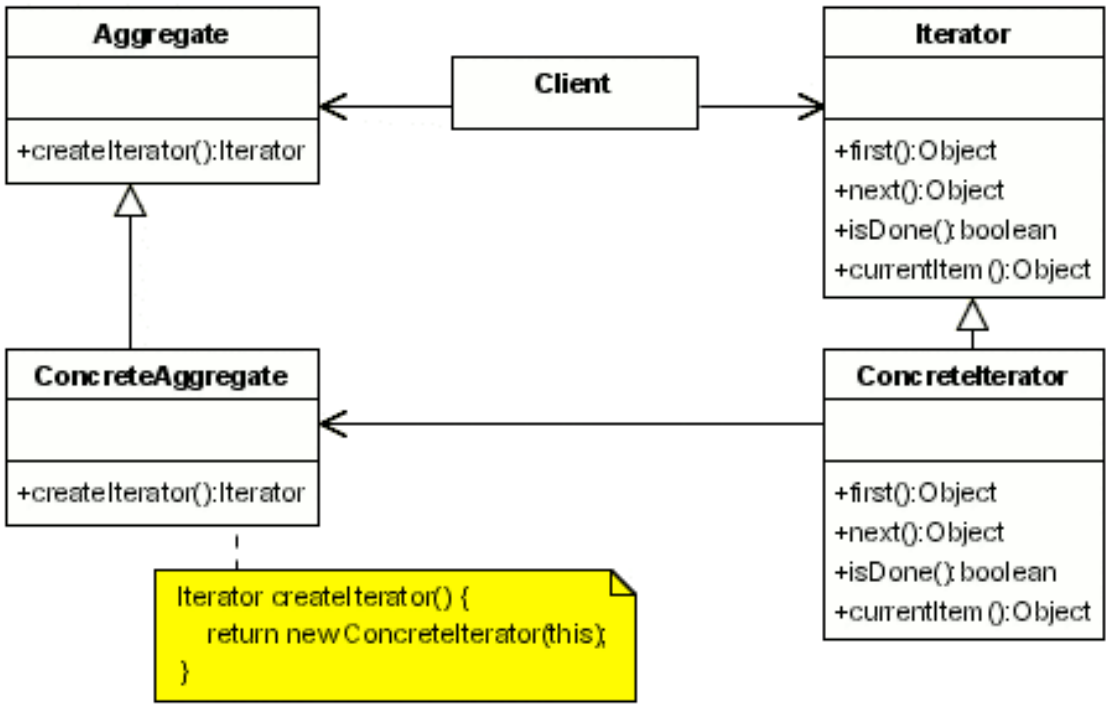
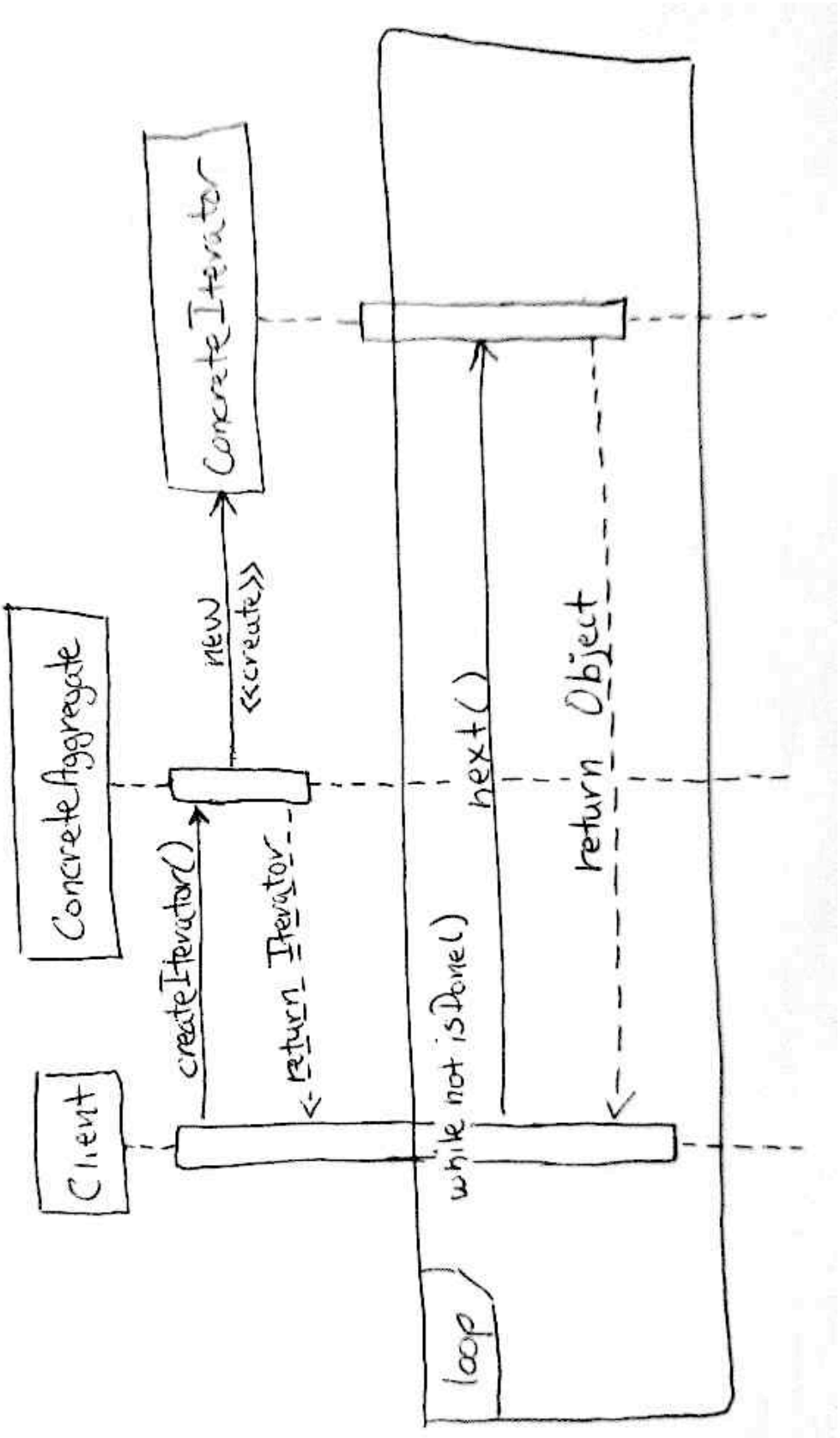


Figure 3: Class Diagram for Iterator Pattern

Marks for: (1) Exactly three objects in diagram (2) Client's call to ConcreteAggregate.createIterator() as solid arrow, and lower than other objects because it didn't exist at beginning (3) Call to ConcreteIterator's constructor with solid arrow, labeled as "new" (or similar), and <<create>> stereotype (4) return dashed line with type Iterator (5) loop interaction frame (6) loop shows end condition (7) calls inside loop show asking for and returning Objects. May also call other Iterator methods, and use other interaction frames as long as it makes sense.

[answer question 2(a) here]



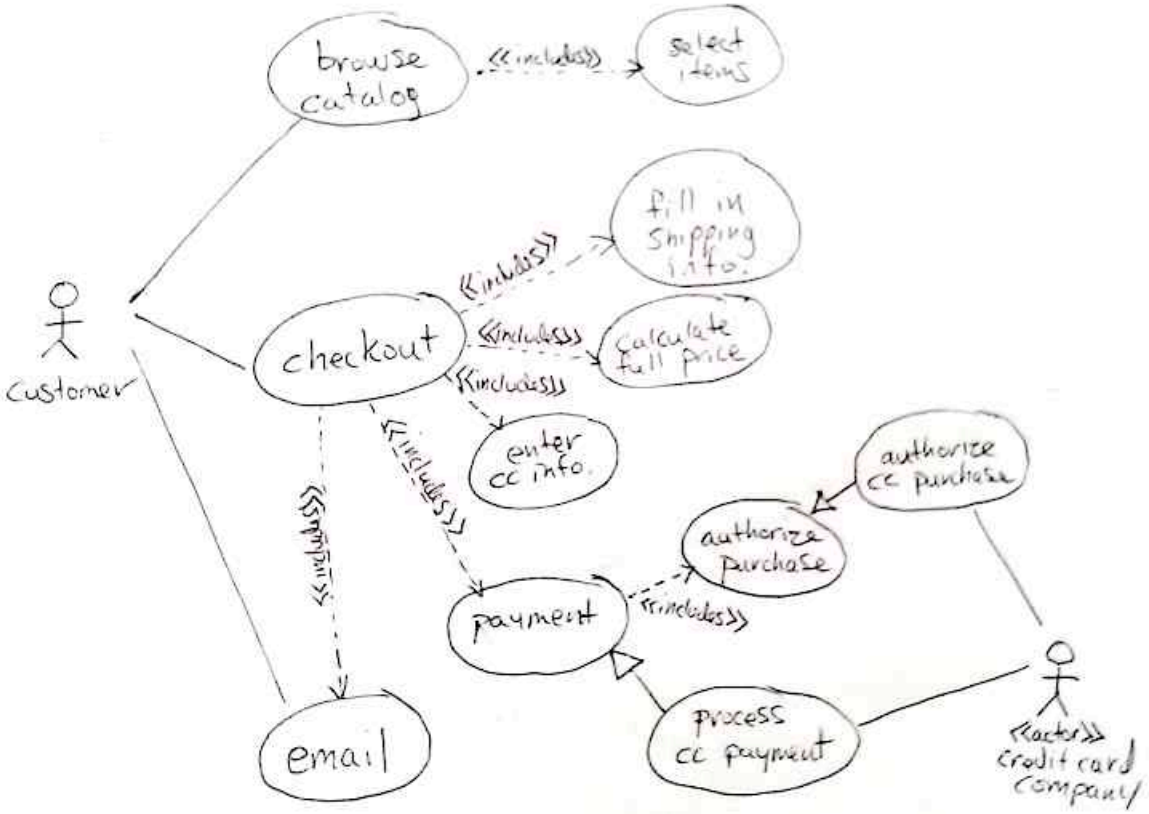
(**b) [Use Case Diagrams – 10 marks]** Draw a UML Use Case diagram for the detailed use case discussed in lecture below. Use page 9 (and the scratch paper at the back of this package, if necessary) to draw the Use Case Diagram.

Use Case – Buy a Product

1. Customer browses catalog & selects items to buy
2. Customer goes to check out
3. Customer fills in shipping information (address, next-day or 3-day delivery)
4. System presents full pricing information
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

[answer question 2(b) here]

One possible solution below. There are many others.



Marks for: (1) completeness: all use cases listed (2) show all the <<includes>> for checkout (3) acknowledge the external <<actor>> representing the credit card company (or equivalent), and (4) optionally show some <<extends>> relationships, and generally (5) it makes some sense.

3. [Software Process, 10 marks Total]
List, and discuss in a few sentences, at least three (3) factors to consider when selecting an appropriate software development process (ex. XP, waterfall, etc.) for a particular software development project.

Listed below are a few sample answers. Many others are also accatpable...

Is the development mainly experimental, or will it become a commercial/open-source product with users other than the developers themselves? If the project is experimental, or a "spike," then no formal process is needed at all. If not, then more thought is required.

Risk factors. Could someone get hurt or die if the software doesn't work correctly? If so, then agile methods may not be as appropriate as a more regimented waterfall-like process. This is not to say that agile methods cannot be used at all, but more care would need to be taken if, for example, Scrum was the chosen process.

Deployment method. One extreme being Software-as-a-Service (SaaS) and the other is a scenario where a skilled install technician brings the software (and possibly the hardware) to the install site, and performs the installation there. Commercial off-the-shelf (COTS) is somewhere between these extremes. Agile methods like XP and Scrum are often the best choice for SaaS because they tend to allow the development team to do more frequent releases.

Do the developers and the stakeholders understand the problem, and potential solution, well? If not, then a process that lets the stakeholders participate in early UAT might be best. Strict Waterfall (or Spiral for first release) may be a bad choice in this case because UAT only happens at the end, when correcting design mistakes is much more expensive.

4. [Verification and Validation (V&V), 10 marks Total]

As discussed in class, Verification and Validation, or V&V, is often referred to as IV&V, where the I stands for Independent. This means that those responsible for V&V are independent from those who are responsible for producing the product. Independence is typically broken down into three categories:

- Managerial Independence
- Financial Independence
- Technical Independence

In the space below, describe each type of independence and why it is important to get unbiased V&V. Also, for each type of independence, give an example of something that can go wrong if you don't have it.

Managerial: The V&V team and the R&D team do not share any reporting structure. This is important so that you don't have a manager in charge of both that will bias one over the other. Without this independence, the R&D manager may cut effort from V&V in favor of adding more features, and quality will suffer.

Financial: R&D and V&V funding comes from totally independent budgets. If they came from the same budget, funds from one could be diverted to the other. If funds are diverted from V&V quality will again suffer.

Technical: The staff performing V&V activities are not the same people as those performing R&D tasks. Further, the V&V staff do not take technical advice or direction from R&D. If there was no technical independence, the R&D team may influence the V&V team's technical decisions, tools they employ, and bug finding strategies. This could increase the probability that bugs hidden to developers would also remain hidden to V&V staff.

[scratch paper]

[scratch paper]

[scratch paper]