CISC 327 - Software Quality Assurance

Lecture 6

Course Project

Course Information

- Tutorials and Advising
  - Project advising @ room/time TBA
    - Advising times will be informal, designed to provide you with practical and technical advice on your project
    - Occasionally a formal tutorial <u>may</u> be scheduled if necessary
  - Online links on the course web page for:
    - Linux command line programming and shell scripting (macOS usually similar)
    - Windows command line programming and batch scripting if you prefer

Course Information

- Assignment Submission
  - Assignments will be handed in as a PDF document in onQ, by 10pm on the due date
  - Be sure to indicate clearly your team name and student names and numbers on **every** submission!
  - This is a course in quality – neatness counts!
  - Think of your submission as a professional paper report, with appropriate titling, sectioning, paging

Course Project

- Project Phases
  - The project will be done in several phases, each of which will be an assignment
  - Phases will cover steps in the process of creating a quality software result in the context of an eXtreme Programming process model
  - Assignments will be on the quality control aspects of requirements, prototyping, testing, integration, and analysis of the product you are building
  - You can usually work ahead on the next assignment in advance to manage your time

Course Project

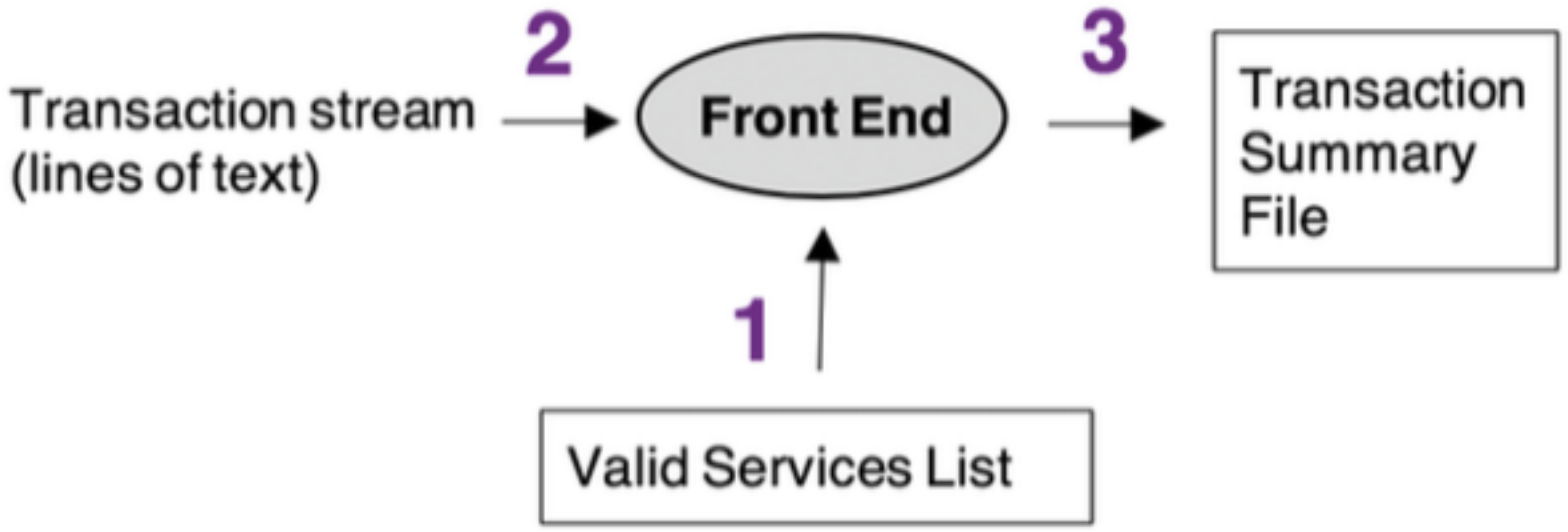
- Project Phases
  - Assignments should always use the simplest possible solution
  - Assignments must be done exactly as the assignment specification says - no exceptions!
  - You can ask the customer (me) for clarifications about the requirements or expectations any time
  - I will respond and post answers quickly

QIES

- Queen's Intercity Excursion System
  - Transportation ticket transaction system
- Consists of a Front End and a Back Office
  - The Front End is a standalone retail sales terminal for ticket transactions
    - login, logout, sellticket, cancelticket, changeticket, plus service creation and deletion when possible
  - The Back Office is an overnight batch processor to maintain and update a Central Services File
    - Aggregate the information from a campus-wide set of Front End terminals

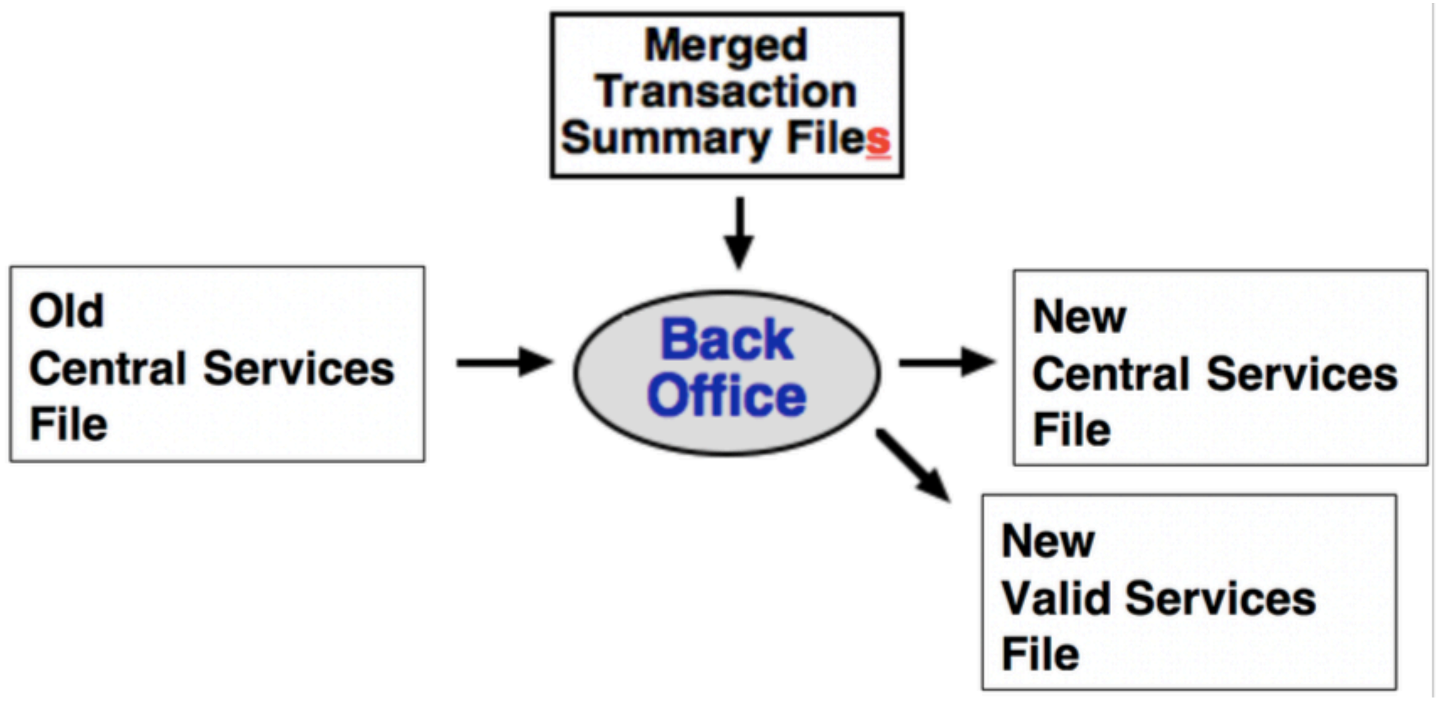
QIES

- The Front End
  - Reads in a list of valid services **(1)**,
    processes a stream of transactions one at a time **(2)**,
    and writes out a summary file of transactions at the end of the day **(3)**



QIES

- The Back Office
  - Reads in the previous day's Central Services file and applies all of today's transactions from a set of transaction files to produce new Central Services and Valid Services files


Merged Transaction Summary Files
Old Central Services File
Back Office
New Central Services File
New Valid Services File

QIES

- The Front End and Back Office **must** be separate!
  - While you **might** be able to combine them into a single program, real projects often have parts that are too large to be combined
  - We want you to get the experience of building and integrating multiple, high-quality programs
  - In this project, the “information split” (e.g. only the Back Office knows capacities and ticket numbers) is somewhat artificial, but real projects do have (better-motivated) information splits

QIES Front End Requirements

- Informal Customer Requirements for the Front End
  - The Front End handles a sequence of transactions, each of which begins with a single transaction code (word of text) on a separate line
  - The Front End must handle the following transactions :

login	start a Front End session (processing day)
logout	end a Front End session
createservice	create a new service (privileged)
deleteservice	delete an existing service (privileged)
sellticket	sell tickets for a service
cancelticket	cancel tickets for a service
changeticket	move sold tickets to another service

QIES Front End Terminology

- "Service"
  - In the context of the project, "service" has a specific meaning: a **trip** (transportation service) for which tickets can be sold
  - For simplicity, all services are "excursions" (e.g. sightseeing tours) that start/end at the same point
    - Ignore issues with transfers between buses, minimum transfer times, route planning, etc.
- "Ticket number"
  - In the context of the project, "ticket number" always means the **number of tickets**, not an "ID number" associated with a specific ticket (which does not exist in the requirements) ("ticket count" might have been better)

QIES Front End Requirements

- What does a sample session look like?
  - Let's say on a given day, a single customer goes to a retail outlet and buys three tickets on Service #10223
  - And then immediately cancels two of the tickets

```
login
agent
sellticket
10223
3
cancelticket
10223
2
logout
```

QIES Front End Requirements

- login: start a Front End session
  - should ask for the type of session, which can be either
    - agent—ordinary retail agent mode
    - planner—privileged (service planner) mode
  - after type is accepted, reads in the valid services file (see requirements) and begins accepting other transactions

QIES Front End Requirements

- Constraints:
  - no transaction other than login should be accepted before a login
  - no subsequent login should be accepted after a login, until after a logout
  - after an agent login, only unprivileged transactions are accepted
  - after a planner (privileged) login, all transactions are accepted

QIES Front End Requirements

- logout: end a Front End session
  - should write out the transaction summary file (see requirements for the file) and stop accepting any transactions except login
  - Constraints:
    - should only be accepted when logged in
    - no transaction other than login should be accepted after a logout

QIES Front End Requirements

- createservice: create a new service
  - should ask for the new **service number, date and time**, and **service name** (as text lines)
  - should save this information for the transaction summary file, but no transactions (e.g. selling tickets) for the new service should be accepted in this session

QIES Front End Requirements

- Constraints:
  - privileged transaction, only accepted when logged in to planner mode
  - new service number is exactly five decimal digits not beginning with 0
  - new service number must be different from all other current service numbers
  - new date and time must be in the format `YYYYMMDD.hhmm` (details in requirements)
  - new service name is between 3 and 39 alphanumeric characters, possibly including spaces but not beginning or ending with a space

QIES Front End Requirements

- deleteservice: delete an existing service
  - should ask for the service number and service name (as text lines)
  - should check that the service number is valid, and save the service number and name in the transaction summary file
  - Constraints:
    - privileged transaction, only accepted when logged in to planner mode
    - no further transactions should be accepted on a deleted service

QIES Front End Requirements

- sellticket: sell a ticket for a service
  - should ask for the service number and the number of tickets (as text lines)
  - should check that the service number and number of tickets are valid
  - should save info for the transaction summary file

QIES Front End Requirements

- canceltlicket: cancel sold tickets
  - should ask for the service number and the number of tickets to be cancelled (as text lines)
  - should check that the service number and number of tickets are valid
  - should save info for the transaction summary file
  - Constraints:
    - no more than 10 tickets can be cancelled **per service** in a single session in "agent" mode (no limit in planner mode)
    - no more than 20 tickets **in total** can be cancelled in a single session in "agent" mode (no limit in planner mode)

QIES Front End Requirements

- changeticket:
  change tickets from one service to another
  - should ask for the current service number, the new service number, and the number of tickets (as text lines)
  - should check that the service numbers are valid
  - should save info for the transaction summary file
  - Constraints:
    - no more than 20 tickets **in total** can be cancelled in a single session in "agent" mode (no limit in planner mode)

QIES Front End Requirements

- Transaction Summary File
  - At the end of each session (processing day), when the logout transaction is processed, a transaction summary file for the day is written, listing every transaction made in the session
  - Contains transaction messages (text lines) of the form:

```
CCC AAAA MMMM  BBBB NNNNNN YYYYMMDDhhmm
```

QIES Front End Requirements

- Transaction Summary File

```
CCC AAAA MMMM  BBBB NNNNNN YYYYMMDDhhmm
```

  - CCC is a three-character transaction code, where SEL = sell tickets, CAN = cancel tickets, CHG = change tickets, CRE = create service, DEL = delete service, EOS = end of session
  - AAAA is the first (source) service number
  - MMMM is the number of tickets
  - BBBB is the second (destination) service number
  - NNNNNN is the service name
  - YYYYMMDDhhmm is the service date and time

QIES Front End Requirements

- Constraints:
  - each line is at most 74 characters (plus newline)
  - the transaction code is always the first three characters of the line
  - items are separated by exactly one space
  - service numbers are always exactly five decimal digits, not beginning with 0 (e.g., 93456, 19450)
  - ticket numbers are between 1 and 4 decimal digits, between 1 and 1000
  - service names are between 3 and 30 alphanumeric/quote characters (A-Z, a-z, 0-9, '), possibly including spaces, but not beginning or ending with a space,
    e.g. KFLA 67 Gananoque '''Stra'n''g'e 'ButAllowed'
  - unused numeric fields are filled with zeros (e.g., 00000 for service numbers, 0 for ticket numbers)
  - unused service name fields are filled with four asterisks: ****
  - the file ends with an end of session (EOS) transaction code

QIES Front End Requirements

- Valid Services List File
  - Consists of text lines each containing only a service number
  - Constraints:
    - each line is exactly 5 characters (plus newline)
    - service numbers are always exactly five decimal digits, not beginning with 0 (e.g., 10327)
    - the file ends with the special (invalid) service number 00000
    - Comes from the Back Office, so you can assume it is well-formed

QIES Front End Requirements

- General Requirements for the Front End
  - The Front End should never crash, and should never stop except as directed by transactions
  - The Front End cannot depend on valid (terminal) input – it must gracefully and politely handle bad input of all kinds
    - But: you can assume that input is at least lines of text!

QIES Back Office Requirements

- Informal Customer Requirements for the Back Office
  - The Back Office reads the Central Services File and the Merged Transaction Summary File (see below)
  - It applies all transactions to the central services to produce the New Central Services File and the New Valid Services List File

QIES Back Office Requirements

- The Back Office enforces the following business constraints, and produces a failed constraint log on the terminal as it processes transactions
  - Constraints:
    - no service should ever have a capacity (number of seats) of 0 or less, nor greater than 1000
    - the number of tickets sold cannot be negative
    - the number of tickets sold cannot exceed the capacity
    - a newly created service must have a new, unused service number, and zero sold tickets
    - a deleted service must have zero sold tickets
    - the service name given in a delete transaction must be the same as the name associated with the deleted service number

QIES Back Office Requirements

- The Central Services File
  - The Central Services File consists of text lines of the form:

AAA CCCC MMMM NNNN YYYYMMDD.hhmm

where:

- AAA is the service number
- CCCC is the service capacity
- MMM is the number of tickets sold
- NNNN is the service name
- YYYYMMDD.hhmm is the service date and time

  - Constraints:
    - each line is at most 68 characters (plus newline)
    - items are separated by exactly one space
    - service numbers, names, and dates-and-times are as described for the Transaction Summary File
    - the Central Services File must always be kept in ascending order by service number

QIES Back Office Requirements

- The Merged Transaction Summary File
  - The concatenation of any number of Transaction Summary Files output from Front Ends, ended with an empty one (one containing no real transactions, just a transaction with an EOS transaction code and unused other fields)
- The New Valid Services List File
  - A file containing every active service number in the New Central Services File, in the format described for the Front End

QIES Back Office Requirements

- General Requirements for the Back Office
  - The Back Office uses only internal files, so it can assume correct input format on all files
  - However, the values of all fields should be checked for validity, and the Back End should stop immediately and log a fatal error on the terminal if any value is invalid

CISC 327 Course Project

- Assignment 1: Front End Requirements Tests
  - Due Friday, October 5th
    - Create and organize a complete set of requirements tests for the Front End of QIES to test for every required behaviour
    - Bonus for discovering missing or erroneous requirements (if customer agrees)
  - Hand in as a PDF file by onQ submission before 10pm on the due date, but you are encouraged to work ahead and hand in assignments early! *(...once we post them)*

CISC 327 Course Project

- You should hand in:
  1. An organized list of all your test cases and what they are intended to test (a table of test names and intentions, in English)
  2. For each test case, the actual test input file and expected output file (as text file printouts)
  3. A test plan document, outlining how your tests are organized (in directories or whatever), how they will be run (as shell scripts, Windows batch files, or whatever), and how the output will be stored and organized for reporting and comparison with later runs (make text file printouts of any directory structures and script files created)

CISC 327 Course Project

- What does a test case look like?

Test T1: login command, agent case
Purpose: check that login is accepted
Input t1in.txt:

```
login
agent
logout
```

Input files: valid services file with no services
Output files: transaction summary file with no transactions
Terminal output t1out.txt:

```
empty, or possibly information messages in response
to commands
```

CISC 327 Course Project

- But first: Assignment #0!
  - Choose teammates to pair program with
  - Think about the programming language and environment you want to work in
  - Sign the team agreement, due Friday in lecture or on onQ
  - If you can't find teammates, email me
- No lecture on Thursday, Sept. 20th
- Friday's lecture: review for Mini-Exam #1
- Mini-Exam #1 is Monday, Sept. 24th