ELEC3225 Applied Programming Concepts

Assignment #2

Problem Statement

The Aquarium system is designed to manage marine animals and staff including trainers, veterinarian technicians, and veterinarians. It includes features for listing animals, managing health data, food tracking, and staff scheduling.

Requirements Engineering

The system requires specification of features such as animal listing, staff roles, veterinary scheduling, and alerts. Feasibility is confirmed using standard technologies like Python classes, databases (SQLite), and basic GUI frameworks.

Requirements Engineering

1. Feasibility Study

- Does the technology exist needed to create the system – Yes. We need classes and objects, a database, and a user interface.
- Does it fit in the budget – yes. We will always assume yes in this course.

Conclusion – the project is feasible, and we can continue.

2. Requirements Elicitation

- We have examined past systems and gathered specifications from users.
- Sources include existing systems and feedback from trainers, vet techs, veterinarians, and management.

3. Requirements Specification

- All users can list animals, filtered by species or country.
- Veterinarians and vet techs can manage health data and receive alerts for upcoming checkups.
- Trainers can manage food and weight records, with alerts for overdue weights.
- Managers can view staff information and auto-generate fair work schedules.

4. Requirements Validation

Requirements are reviewed by group members to ensure accuracy and completeness.

Design and Implementation
- Architectural design: classes/objects, functions, DB, and UI.
- Interface design: connection of UI with logic and database.
- Component design: classes for staff, animals, UI, etc.
- Database design: Staff and Animal tables, with utility functions for age and food.

Software Validation
- Component testing: unit tests for each module.
- System testing: full integration testing.
- Acceptance testing: tested with actual or realistic data.

Software Evolution
Modifications over time based on user feedback, changing needs, and bug fixes.

Software Process Models

Waterfall Model
This model follows a linear sequence where each phase is completed before the next begins. This model is well structured, but lacks flexibility. This is best for when projects have fixed requirements.

1. Requirements Analysis (Week 1): Define all features and functionality for users and animals.
2. System Design (Week 2): Define database schema, user roles, alerts, and interface components.
3. Implementation (Week 3-4): Code the system including all classes and interfaces.
4. Testing (Week 5): Perform component, system, and acceptance testing. Fully integrate system elements together and start testing using "real" data.
5. Deployment and Maintenance (Week 6): Finalize UI, fix bugs, and prepare for future changes. Deploy the "final" version.

Use Case: Best suited for projects with clearly understood and unchanging requirements.

Incremental Development Model
This model builds the system in iterative steps, with working versions delivered after each increment. This allows for flexibility to modify requirements as the project progresses.

Iteration 1 (Week 1): Implement animal and staff listing functionality. This helps deliver early, basic functionality.
Iteration 2 (Week 2): Add vet/vet tech features like checkup scheduling and health record updates. Iteration 3 (Week 3): Add trainer features for managing food and weight data.

Iteration 4 (Week 4): Implement alerts for overdue checkups and weight logs.

Iteration 5 (Week 5-6): Add management scheduling interface and finalize integration/testing.

Use Case: Good for evolving requirements and early delivery of partial systems. This model allows for early delivery of basic functionalities.

Integration and Configuration Model

This model emphasizes reuse of existing software tools and libraries.

Component Selection (Week 1):
- Use SQLite3 (https://www.sqlite.org) for lightweight database.
- Use Tkinter (https://docs.python.org/3/library/tkinter.html) for GUI.
- Use datetime and smtplib for alerts/reminders.

Configuration and Integration (Week 2-3):
- Connect database to Python backend logic.
- Build user roles and login authentication.

-Start setting up the User Interface and connecting it with the backend logic

Development (Week 4-5):
- Customize user workflows (animal tracking, vet tools, alerts, staff scheduling).

Testing and Refinement (Week 6):
- Final system validation and bug fixes.

-Conduct full system testing

Use Case: Efficient where trusted components already exist and need adaptation. This helps reduce time and reduce custom programming.

Conclusion

Each process model offers distinct advantages. Waterfall works best when requirements are fixed, incremental supports evolving systems with early feedback, and integrate/configure reduces development time through reuse. The Incremental model is better for systems that will change over time, and the Integration and Configuration model is better for building quick code by using existing tools. The Aquarium system is ideal for either the incremental or integration model, depending on availability of components and development speed goals.