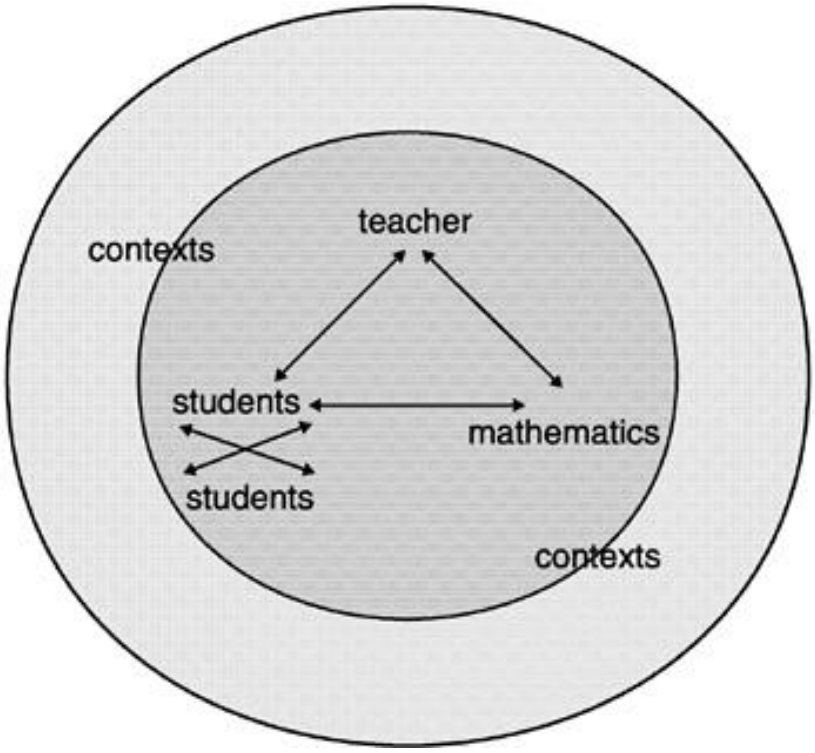
Computer Programing

Activity 1

Part 1 Evaluation of application programming interfaces (API)

A creative problem-solving program will be developed by keeping in mind different skills so that the child will better understand the course. Three areas of learning are essential to the teaching of school mathematical, mathematics knowledge, pupil knowledge and teaching. Mathematics and learners are also the number of edges of the triangle, and educational tasks are the relationships defined by bows.



An API is typically similar to a software library, The API specifies and recommends the behaviour that the library plans for the actual execution of this set of rules. A single API may have numerous implementations in various libraries that share the same programming interface (or none, being abstract). A software framework may be related to an API, too a framework could be based on various libraries that support multiple APIs, however the content can be

broadened to include a new class inserted into the framework itself in contrast to the standard usage of an API. API is used for compatibility. Furthermore the total control flow of the programme, by control reversal or related processes, may be out of caller command and in structure (Ofoeda, Boateng, & Effah, 2019).

APIs that are meant for use in upwards of one word have the capability to dynamically map an API to functions which are more natural or augmented with (syntactic or semanticized) in certain languages. It is known as the linking of languages, which is an API itself. The goal is to encapsulate most of the API's necessary functionality, leaving a thin" layer suitable for each language. API design is important because it is intended to be written once and used several times, and due to compatibility problems, subsequent changes can affect users. Complex APIs can be challenging, and difficult use can deter acceptance as it increases the demand for professional programmers who can use it efficiently and effectively. In Henning's work, an example of the implications of an excessively complex API can be found.

The fundamental concept behind integrating these study methods is to have the same experiment with multiple viewpoints to interpret the effects fully. SIM provides insights into sign systems and notations that the creator uses to articulate his vision. The basis for a cognitive review of the experiments is given by CDN, which is important since a new API implies a programmer's learning experience. By providing additional evidence to support or refute other findings, these two methods can complement discourse analysis (Afonso, Cerqueira, & Souza, 2012).

A fascinating feature of API design is that they may have several distinct and particular objectives that typically make them unique. Behind it, there might be subtleties that render those choices crucial to its completeness, accessibility, and versatility. The design process should

expect some critical aspects of API use, affecting the underlying device design. A programmer should have a good mental model of the software objects being reused while designing software to correctly adapt these models to his design and name the operations and services available accordingly. If the abstractions presented are not well known, this may lead to subtle mistakes that might occur later in the software life cycle (Meng, Steinhardt, & Schubert, 2018).

Part 2 Appraisal of the stages of the software development life cycle

SDLC is a systemic framework for software design that maintains the consistency and the accuracy of the software developed. The SDLC approach aims to deliver high-quality consumer applications. The design of the method should be completed within the predetermined time and cost structure. SDLC consists of a detailed schedule detailing the specification, implementation and management of relevant applications. The process and deliverables that feed into the next step are in every stage of the SDLC life cycle. SDLC stands for the Life Cycle of Software Development and is also known as the life-cycle of application development.

The current framework is analyzed during the review process, and new demands are defined and organized according to their interrelationships. Alternative design strategies are developed after the determination of requirements. The selected alternative design strategy is translated into conceptual and physical design requirements during design. The information system is programmed, checked, and implemented in the company during installation and service. The system is systematically repaired and enhanced until the system is put into production. (P., 2016)

A life cycle for system development is a series of stages that mark an information system's development.

1. **Requirement Analysis**

   In this step, software development needs in the requirements review process of SDLC (Software Development Life Cycle). This move is structured to compile all project information or even the requirement analysis process is meant to collect the specifics of each requirement and also to ensure everyone knows the essence of the job and how each requirement is fulfilled.

2. **Design**

   The next step of the life cycle seems to be the design stage. The software and machine high-level design is initiated by engineers and number of technology during the development process to satisfy any requirement. Different factors are analysed such as risk, technologies to hire, team capability, project constraints, time and budget. The partners discuss concept technical issues and analyse them. The best production approach is then selected for the product (Kumar & Dubey, 2013).

3. **Implementation**

   It is the stage where all the specifications that are obtained from the client are enforced. In these stages, coding is started as required by the customer. All begins operating as a data base manager at this point and starts coding the function by making database programmers begin. Project modules and lead developer figures can be said as per programme specifications to construct an integrated interface.

4. **Testing**

   Until the product is shipped to clients, testing is the last step of the Software Development Life Cycle. In this step, we verify that our program works or not according

to our expectations. We also verify that SRS completely fulfills the customer's entire requirement at the time of the agreement.

5. **Deployment and Maintenance**

After software is developed, we will use the customer-based software and normally a supporting team can manage all issues in post-production. Assume that a manufacturing issue and how bad it is is told to the implementation team.

The organization's overall information systems needs are evaluated and prioritized during planning and selection. A possible information systems project is established, and a case is made to continue or not continue with the project (Rather & Bhatnagar, 2015).

Activity 2

Language Constructs for Solving Math Problems

Programming is to take an algorithm and encode it into a language, a programming language, to run it on a computer. While there are many languages and several different machine styles, the first critical step is the solution. No software can be found without an algorithm. The

analysis of programming is not computer science. However, coding is an essential aspect of what a piece of information does. The way we make representation for the solutions is always programming. This portrayal of the language system it is formed thus become a central part of the discipline. Algorithms define the solutions to problems in terms of the information available for the prediction problem and the steps taken to obtain the desired outcome. The programming languages should provide a description of the method as well as the data. Languages have control constructions and data types for this purpose.

Control structures allow for the convenient yet unambiguous description of algorithmic measures. At a minimum, algorithms require serial processing systems, decision selection, and repeated control iteration. Unless these essential statements are included in the vocabulary, it can be used to represent that algorithm. Both data objects are displayed on the screen as binary digit strings. We must have data types to assign these sequences meaning. Data types offer an interpretation of these binary data to consider the data in terms that make sense of the problem becomes solved. The low-level integrated data types (sometimes referred to as primary data types) form the building blocks for algorithms' development. For one, most programming languages have an integer type of data. The binary digits sequences in the machine's memories can be translated as integrals and the usual definitions common to integers (e.g., 23, 654, and -19). Furthermore, a data type defines the operations in which data objects can participate.

Besides, integers are popular for operations like addition, subtraction as well as multiplication. They have been expected to engage in these arithmetic operations in numerical data. The challenge for us also lies in the very complexity of problems and remedies. Those simple linguistic constructions and data kinds are normally inconvenient when humans work

with the problem-solving process, although adequate to provide complex solutions. This uncertainty has to be handled, and strategies are developed (Bernardo, 2002)

Activity 3

Pseudocode

An algorithm is a tool for addressing a dilemma about the action to be done and the exercise order. The series of steps taken to fix a problem is just an algorithm. The measures are usually a case-type declaration and are "sequence," "choice," "installment." "Sequence declarations" in C are essential. The "selection" is the statement "if so and a variety of statements like "while," "do," and also for" satisfy iterations, as well as the statement of the case-type, is fulfilled by the statement "turn." Pseudocode is an interface that enables programmers to build algorithms. Pseudocode is an algorithmic (text-based) method of design. The pseudo-code laws are simple. Both "dependency" sentences must be indented (Yulianto, Prabowo, Kosala, & Hapsara, 2018). This involves, do, and alter, if. This principle is explained by the descriptions below.

The total will be set to zero

The grade counter will be set to one

Where the grade counter is lesser than or considered equal to ten

Inputting the next grade

Adding the grades into the whole

Setting the overall class dividend by ten

Printing the calculation

Activity 4 - Test Results

Test Scenario: Many mistakes derive from people's errors and inaccuracies in the source code, mathematical formulae, and arithmetical measurement of a program. For various program faults, flaws in the estimation of too many defects remain the biggest concern. Checking is also very necessary as end-user dislike program errors. If the measurement error is not fixed, major consequences will be dealt with.

Expected Results: The expected result is determined by multiplying each future result by each event's likelihood and summing all these effects in statistics and probability analysis. Investors should select the situation that will most likely yield the desired outcomes by estimating predicted values. The ascertained outcomes in this particular case are about the optimum implementation of the pseudocode and its effective results that can be found. Here the results were equal to ten that was quite approachable.

Actual Results: The results that were obtained were nine that was below the expected outcome. Thus there will be a gap found that can be overcome by the effective computer programming and continuous development of computer programming software.

Pass/Fail: It can be found that computer programming is producing effective results. Some modifications are needed, and the design will be efficient and effective.

Recommendations

- The representatives of higher education institutions of rapid growth in computer science must take deliberate measures to counter this trend urgently.

- From tailored admission controls to supplementary support for the demand to much more comprehensive structural improvements that spread over the IT department, several acts can be viewed as an integral part of integrated process innovation.
- To minimize the increasing workload of faculty and personnel in informatics and associated organizations and constraints coming from underprivileged facilities, institutions facing the growth of computer science registration need to take severe account of the surge in funding.
- Any organization may deem it appropriate or unavoidable to place restrictions on enrolment in computer science and related courses. However, the university community's implications and advantages and expenses should be weighed carefully before enforcing restrictions on courses and major enrollments.
- Institutional leadership must directly assist in developing appropriate faculty and faculty-size priorities and developing initiatives to increase retention of faculty in the IT departments or services. There should be significant consideration given to raising the size and position of the university faculty

---

References

Afonso, L., Cerqueira, R. F., & Souza, C. (2012). Evaluating application programming interfaces as communication artefacts. *Psychology of Programming Interest Group*, 1-12.

Kumar, S., & Dubey, P. (2013). Software Development Life Cycle (Sdlc) Analytical Comparison And Survey On Traditional And Agile Methodology. *National Monthly Refereed Journal Of Research In Science & Technology, 2*(8), 22-30. Retrieved from

https://www.researchgate.net/publication/319716548_SOFTWARE_DEVELOPMENT_LIFE_CYCLE_SDLC_ANALYTICAL_COMPARISON_AND_SURVEY_ON_TRADITIONAL_AND_AGILE_METHODOLOGY

Meng, M., Steinhardt, S., & Schubert, A. (2018). Application Programming Interface Documentation: What Do Software Developers Want? *Journal of Technical Writing and Communication, 48*(3), 295–330. Retrieved from https://www.researchgate.net/publication/318733467_Application_Programming_Interface_Documentation_What_Do_Software_Developers_Want

Ofoeda, J., Boateng, R., & Effah, J. (2019). Application Programming Interface (API) Research: A Review of the Past to Inform the Future. *International Journal of Enterprise Information Systems, 15*(3), 76-95. Retrieved from https://www.researchgate.net/publication/334145268_Application_Programming_Interface_API_Research_A_Review_of_the_Past_to_Inform_the_Future

P., M. G. (2016). Software Development Lifecycle Model (Sdlc) Incorporated With Release Management. *International Research Journal of Engineering and Technology (IRJET), 3*(4), 1536-1543. Retrieved from https://www.irjet.net/archives/V3/i4/IRJET-V3I4304.pdf

Rather, M. A., & Bhatnagar, M. V. (2015). A Comparative Study Of Software Development Life Cycle Models. *International Journal of Application or Innovation in Engineering & Management, 4*(10), 23-29. Retrieved from https://www.researchgate.net/publication/305863548_A_comprative_study_of_sdlc_model

Bernardo, A. B. (2002). Language and Mathematical Problem Solving Among Bilinguals. *The Journal of Psychology Interdisciplinary and Applied, 136*(3), 283-297. doi:10.1080/00223980209604156

Yulianto, B., Prabowo, H., Kosala, R., & Hapsara, M. (2018). Novice programmer = (sourcecode) (pseudocode) algorithm. *Journal of Computer Science, 14*(4), 477-484. doi:10.3844/jcssp.2018.477.484

Quigley, A. (2011). Welcome to Computers—A New Open Access Journal for Computer Science. *Computers*, *1*(1), 1-2. doi:10.3390/computers1010001