Name: Shravani Sandesh Kulkarni
Student Id- 40272412

EXERCISES - CHAPTER 1 & 2

Q.1.1) It is said that government spending on IT is increasing as government departments take initiatives to improve customer service or have a wider reach of services. Find out what factors are responsible for the increase of IT spending by government agencies. Also, list and explain the three biggest IT projects undertaken by the federal government in recent times?

-> As we are moving in an era of digitalization, governments worldwide are transitioning from traditional procedures to digital setups. This initiative is taken to enhance their services, provide better services and improve customer services, hence the spending of federal governments on IT has increased drastically over the past few years.
This spending can be attributed to several factors. Some key factors include:

1) **Digital Transformation Initiatives:** To modernize their operations and service delivery, governments all over the world are undergoing digital transformation. This entails integrating digital technologies into all facets of government operations, which raises the need for IT spending.

2) **Improved Service Delivery:** Governments aim to provide better services to citizens and businesses. Investing in IT allows for the development of user-friendly online platforms, mobile applications, and automation tools to streamline processes and deliver services more efficiently.

3) **Cybersecurity Concerns:** The rising threat of cyberattacks has prompted governments to allocate more resources to enhance cybersecurity measures. This includes investments in advanced cybersecurity technologies, training programs, and the development of secure IT infrastructures.

4) **Remote work and Telecommuting:** The COVID-19 pandemic accelerated the adoption of remote work, prompting governments to invest in IT infrastructure and tools to support a distributed workforce. This includes investments in secure remote access, collaboration platforms, and digital communication tools.

5) **Regulatory Compliance:** Governments need to comply with various regulations related to data protection, privacy, and accessibility. Investing in IT systems helps ensure compliance with these regulations, avoiding legal issues and improving public trust.

Following are the biggest IT projects undertaken by the government of Canada in recent times:

1) **E-Procurement Solution (EPS)**: [*Department*: Public Services and Procurement Canada]
Description: EPS is a public sector electronic procurement service that will provide modern and innovative e-tools and applications for all facets of the procurement process including e-sourcing, contract lifecycle management, spend analysis, supplier relationship management and e-purchasing through catalogues. It will also provide one portal for all acquisition needs, facilitate suppliers' interaction with the GC and provide greater accessibility for public sector clients to procure goods and services at the best value possible.
Latest Budget (2022):$144,135,089
Latest Estimated Completion Date (2022): June 30, 2023
It is running behind schedule

2) **Offender Management System (OMS) Modernization [**Department: Correctional Service Canada]
Description: Replace CSC's OMS business processes and system with a modern and supported platform.
Latest Budget (2022):$77,400,000
Latest Estimated Completion Date (2022): February 17, 2027

3) **Latest Estimated Completion Date (2022) [**Department: Canada Revenue Agency]
Description: The objective of the project is to create a sustainable, usable portal model which is flexible enough to maintain current capabilities while accommodating future growth, and technological advancements.
Latest Budget (2022): $52,848,000
Latest Estimated Completion Date (2022): March 31, 2025

Q.2.2) Go to some open-source projects and find out about their project charters. Find out why they have those project charters.

-> Following are some of the open-source projects,
Many open-source projects have project charters or similar documents that outline their goals, governance, and guiding principles.

1) **Rust:** The governance structure is documented in Rust RFC #1068. The effective governance process grows organically over time without being entirely codified as RFCs, especially in the case of day-to-day operation details. One example is the formation of Domain Working Groups in February 2018.

-Project choice

There are many open-source projects out there, but it will be most fruitful to survey those which are similar enough to CPython on a couple of key metrics:
1)The number of contributors and their activity (there are scaling issues that don't make the governance models of very small projects very enlightening for our purposes) ;
2)Being mostly or partly community-driven (company-driven projects can afford different governance options since the company hierarchy has power over the main participants) ;
3)Being faced with important design decisions that require a somewhat formal decision process.

-Regular decision process

Primary work happens via GitHub issues and pull requests. Approving a pull request by any team member allows it to be merged without further process. All merged pull requests end up in the next stable version of Rust.
Notifying relevant people by mentions is important. Listening to the firehose of e-mails for all GitHub activity is not popular.
There are planning and triage meetings open to the public happening on IRC and Discord. They are not very popular because most of the work happens through GitHub. Discussions also happen on official Rust forums (https://users.rust-lang.org/ and https://internals.rust-lang.org/). There is a dedicated moderation team responsible for taking notes and enforcing a code of conduct.

-Key people and their functions

In the Rust project, there are teams responsible for certain areas. For language features there is a "lang team", for tooling, there are "dev tools" and "Cargo", and so on. Contentious issues have facilitators to drive discussion who often aren't the decision-makers. Typically the facilitators are authors of the proposed changes (see "Controversial decision process" below). They ensure all key decision makers are involved along with interested community members. They push towards an agreeable outcome via iteration.
In practice, this means decisions are rarely escalated to the core team.
The most common role of a contributor is team membership. Issue triage/code review privileges without team membership is rare. Contributors have full commit access, code ownership separation is based on trust. Writing to the compiler repository is frowned upon, all changes go through pull requests and get merged by an integration bot after they are reviewed and approved.
New team members are added by nomination by an existing team member.

2) **Jupyter:** The governance structure is documented in the Main Governance Document within the Jupyter Governance repo. The effective governance process grows organically over time as the needs of the project evolve. Formal changes to the Governance Document are submitted via Pull Request, with an open period for comments. After the open period, a Steering Council may call for a vote to ratify the PR changes. Acceptance requires a minimum of 80% of the Steering Council to vote and at least 2/3 of the vote must be positive. The BDFL can act alone to accept or reject changes or override the Steering Council decision; though this would be an extremely rare event.

-*Key people and their functions*

The key people in Jupyter%E2%80%99s Governance are the BDFL, Fernando Perez, and the Steering Council. Contributors can be given the special status of the core contributor. Some may also be Institutional Contributors, who are individuals who contribute to the project as part of their official duties as an Institutional Partner. Fernando Perez, the project founder, is the current and first BDFL. The BDFL may serve as long as desired. The BDFL succession plan is described in the Main Governance Document. In summary, the BDFL may appoint the next BDFL. As a courtesy, it is expected that the BDFL will consult with the Steering Council. If the BDFL can not appoint a successor, the Steering Council will recommend one. Core contributors are individuals who are given rights, such as committing privileges, to act in the best interest of the project within their area of expertise or subproject. An existing core contributor typically recommends someone be given core contributor rights by gathering consensus from project leads, who are experienced core contributors as listed in the README of the project repo. To be recommended and invited as a Steering Council member, an individual must be a Project Contributor who has produced contributions that are substantial in quality and quantity and sustained over at least one year. Potential Council Members are nominated by existing Council members and voted upon by the existing Council after asking if the potential Member is interested and willing to serve in that capacity.

-Regular decision process

Project Jupyter is made up of several GitHub organizations and subprojects within those organizations. Primary work happens via GitHub issues and pull requests. Approving a pull request by any team member allows it to be merged without further process. All merged pull requests end up in the next stable release of a subproject. There is a weekly, public Project-wide meeting that is recorded and posted on YouTube. Some larger GitHub organizations, which are subprojects of Project Jupyter, e.g. JupyterLab and JupyterHub, may have additional public team meetings on a weekly or monthly schedule. Discussions occur on Gitter, the Jupyter mailing list, and most frequently an open issue and/or pull request on GitHub.

3) **Kubernetes:** The Kubernetes community adheres to the following principles:

a) Open: Kubernetes is open source. See repository guidelines and CLA, below.
b) Welcoming and respectful: See the Code of Conduct.
c) Transparent and accessible: Work and collaboration should be done in public. See SIG governance, below.
d) Merit: Ideas and contributions are accepted according to their technical merit and alignment with project objectives, scope, and design principles.

-Code of Conduct

The Kubernetes community abides by the Kubernetes code of conduct.
As contributors and maintainers of this project, and in the interest of fostering an open and welcoming community, we pledge to respect all people who contribute through reporting issues, posting feature requests, updating documentation, submitting pull requests or patches, and other activities.
As a member of the Kubernetes project, you represent the project and your fellow contributors. We value our community tremendously and we'd like to keep cultivating a friendly and collaborative environment for our contributors and users. We want everyone in the community to have positive experiences.

-Scope

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.
The name Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation results from counting the eight letters between the "K" and the "s". Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

-Design Principles

Principles to follow when extending Kubernetes.
API:

1) All APIs should be declarative.
2) API objects should be complementary and composable, not opaque wrappers.
3) The control plane should be transparent -- there are no hidden internal APIs.
4) The cost of API operations should be proportional to the number of objects intentionally operated upon. Therefore, common filtered lookups must be indexed. Beware of patterns of multiple API calls that would incur quadratic behaviour.
5) Object status must be 100% reconstructable by observation. Any history kept must be just an optimization and not required for correct operation.
6) Cluster-wide invariants are difficult to enforce correctly. Try not to add them. If you must have them, don't enforce them atomically in master components, that is contention-prone and doesn't provide a recovery path in the case of a bug allowing the invariant to be violated. Instead, provide a series of checks to reduce the probability of a violation, and make every component involved able to recover from an invariant violation.
7) Low-level APIs should be designed for control by higher-level systems. Higher-level APIs should be intent-oriented (think SLOs) rather than implementation-oriented (think control knobs).

- These open-source projects often have project charters for several reasons:

1) **Define Project Purpose:**
A project charter outlines the fundamental reason for the project's existence.
It includes a mission statement, describing the overarching goal the project aims to achieve.
This clarity helps contributors align their efforts with the project's core objectives.

2) **Establish Governance:**
The charter defines the project's governance structure, specifying roles and responsibilities.
It outlines decision-making processes, ensuring that contributors understand how key choices are made.
This structure helps maintain order and transparency within the project community.

3) **Scope and Objectives:**
The charter delineates the project's boundaries and what falls within its scope.
It outlines specific objectives, providing a roadmap for contributors to follow.
This prevents scope creep and sets realistic expectations for project outcomes.

4) **Community Guidelines:**
Included in the charter are guidelines for community participation, shaping the behaviour and interactions among contributors.
Communication norms and a code of conduct may be specified, fostering a positive and collaborative community atmosphere.
These guidelines contribute to the creation of a welcoming and inclusive project environment.

5) **Stakeholder communication:**
The charter serves as a communication tool, summarizing essential project details for both internal and external stakeholders.
It provides a quick reference point for understanding the project's purpose, governance, and key guidelines.
This aids in stakeholder engagement and ensures a shared understanding of the project's principles and direction.

References:

1) https://github.com/kubernetes/design-proposals-archive/blob/main/architecture/principles.md
2) https://www.itbrew.com/stories/2022/12/21/governments-will-beef-up-their-it-spending-next-year-according-to-gartner-projections
3) https://large-government-of-canada-it-projects.github.io
4) https://peps.python.org/