Software de control per a un nanosatèl·lit

Emilio Garcia

Resum—L'Institut d'Estudis Espacials de Catalunya i l'Institut de Ciències del Espai, actualment estan realitzant un projecte anomenat C3SatP que consisteix en el disseny i prototipatge d'un nanosatèl·lit de fabricació pròpia que permeti donar solucions adaptades a possibles necessitats dins el territori de Catalunya. L'abast d'aquest treball consistia en realitzar un desenvolupament de software. Concretament, el software del nucli d'un dels components d'aquest nanosatèl·lit, anomenat On Board Data Handler, i un mòdul per el software base de tota la plataforma, anomenat C3SatP-Basis. La metodologia per dur a terme aquests objectius, consistia en realitzar reunions cada dues setmanes, on s'establien els requisits a mesura que anaven apareixent, i en paral·lel realitzar un document d'especificació de requisits. Posteriorment, el disseny de les funcionalitats utilitzant diagrames de classes, diagrames de seqüencia i de casos d'us. Per últim, la implementació del software i els casos de test que garantien el compliment dels requisits, en llenguatge C i C++ respectivament, i utilitzant control de versions. Per a la realització del projecte ha calgut l'aprenentatge d'eines i tecnologies diverses, com per exemple IDEs, utilització de llibreries externes i càrrega llibreries dinàmiques. Com a resultat de tot el procés d'enginyeria, s'han aconseguit gran part dels objectius que s'havien plantejat a l'inici.

Paraules clau— C, C++,CubeSat, GoogleTest, Linux, Nanosatel·lit, Software, TDD

Abstract—The Institute of Space Studies of Catalonia and the Institute of Space Sciences are currently carrying out a project called C3SatP which consists in the design and prototyping of a nanosat of its own manufacture that allows to provide adapted solutionsto possible needs within the territory of Catalonia. The scope of this work is, mainly, software development, following an engineering process, of one of the components of this nanosatellite, called On Board Data Handler. In addition, it has also been requested to develop a state machine for the base software of the entire nanosatellite. The methodology consisted in meeting with the tutor every two weeks, where the requisites were established as they appeared, in addition to making a specification requirements document. Later, the design of the functionalities using class diagrams, sequence diagrams and user cases. Finally, the implementation of the software and the test cases that guaranteed compliance with the requirements, in C and C ++ language respectively, and using version control. To carry out the project, learning different tools and technologies have been needed, such as IDEs, use of external libre. As a result of the entire engineering process, most objectives which they had been considered at the beginning, have been achieved.

Index Terms— C, C++,CubeSat, GoogleTest, Linux, Nanosat, Software, TDD.

1 Introducció

En l'àmbit de l'espai, les investigacions que s'han fet al llarg del temps requereixen una infraestructura important per dur a terme les investigacions de cadascun projectes els quals intervenen a cada llançament. Això és causat, en gran part, pels objectius que es volen assolir en les respectives missions, com per exemple anar fins a Mart i prendre certes mesures es necessiten coets amb garanties d'èxit.

Des del punt de vista econòmic, el cost de desenvolupament d'aquestes infraestructures es molt elevat. Moltes entitats que s'encarreguen en fer investigacions en les quals no es necessària una infraestructura tan gran, no són capaces d'assumir aquest cost. No és necessària perquè les mesures que es necessiten per fer els estudis es prenen durant un període curt de temps, si les comparem amb les investigacions convencionals. Davant d'aquesta nova necessitat, surt el terme satèl·lits petits [1] que agrupa tots els satèl·lits amb un pes inferior a 500Kg. Dintre d'aquesta classificació es troben satèl·lits de diferent pes. Els zeptosatèl·lits son els més petits (0.1g) i els minisatèl·lits els més grans, (100 - 500 Kg). En aquest rang es troben els nanosatèl·lits, els quals poden pesar entre 10 i 100 Kg. Les especificacions d'aquest tipus de satèl·lit van ser creades en 1999 a California, per promoure la realització de projectes espacials en entorns acadèmics.

Específicament, existeixen diferents tipus de nanosatèl·lits:

Figura 1. PocketQube

Figura 2. Cubesat

Figura 3. TubeSat

- *E-mail de contacte: emilio.garciaq@e-campus.uab.cat*
- *Menció realitzada: Enginyeria del Software.*
- *Treball tutoritzat per Lluís Gesa Bote (ICE-CSIC)*
- *Curs 2018/19*

Els PocketQube (Figura 1) son semblants estructuralment als CubeSats (Figura 2), però més petits i la aplicació usual es en l'educació ja que també son més econòmics.

Els TubeSats (Figura 3) dissenyat per Interorbital systems no son tan populars degut a que el coet on aniran aquest dispositius encara esta en fase de test [2] i el seu disseny es més un kit que no pas un disseny obert amb unes especificacions mínimes com és el CubeSat [3].

1.1 Estat del art

Avui dia, les agencies espacials d'arreu del mon ja estan realitzant projectes que involucren aquests tipus de infraestructures, com per exemple, la NASA, el passat 26 de novembre de 2018, a través del projecte "Mars Cube One" (MarCO) [4], ha desplegat dos CubeSats com a càrrega del coet que ha portat l'InSight Lander a Mart. Un d'aquests com a copia de seguretat i l'altre per recollir dades del descens de la nau primària.

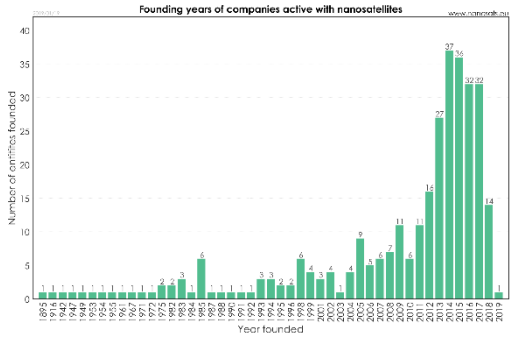


Figura 4. Entitats creades en el mercat de nanosatèl·lits

Com es pot veure a la figura 4, la tendència des del 2005 es que cada cop més, les empreses que es creen siguin enfocades en aquest àmbit. No només això, sinó que els llançament d'aquest tipus de sistemes son més freqüents:

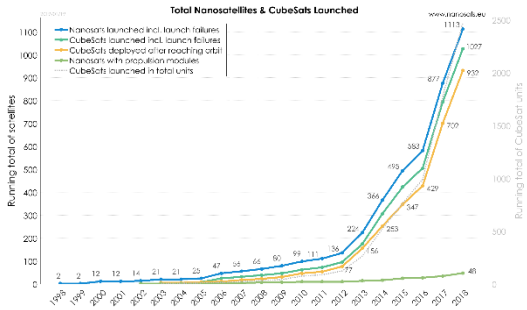


Figura 5. Nanosatèl·lits i CubeSats llançats

A l'àmbit local, l'Institut d'Estudis Espacials de Catalunya (IEEC) i l'Institut de Ciències del Espai (ICE-CSIC) actualment estan realitzant un projecte anomenat **C3SatP** dins el programa "Producte" de la Generalitat en el que consisteix en el disseny i prototipatge d'un nanosatèl·lit de fabricació pròpia que permeti donar solucions adaptades a possibles necessitats dins el territori de Catalunya. Concretament, el prototipatge es un CubeSat, esta format pels següents components:

On Board Computer (ordinador de control del nanosatèl·lit) es basat en un microcontrolador STM32F446RE, el qual disposa d'un ARM Cortex-M4 a 180 Mhz i 512Kb de memòria flash. Com a sistema operatiu s'utilitza FreeRTOS el que es s'anomena, Real Time Operative System o sistemes operatius en temps real, ja que es necessària una actuació immediata.

Com a segon component, tenim l'**On Board Data Handling** (OBDH), és l'ordinador de processament de dades. Respecte a les especificacions hardware conté una FPG-MPSoC que integra un Zynq UltraScale+ MPSoC de 64 bits, que està format per un dual-core Cortex TM-A53 i un dual-core Cortex TM-R5 de processament en temps real. El sistema operatiu que gestionarà L'OBDH esta basat en Linux. Com a sistema d'emmagatzematge conté una memòria SD de 64 Gb.

D'altre banda es compte amb un component extern que és la **Mission Control System** (MCS) que es l'encarregada de enviar comandes i rebre dades del CubeSat definits anteriorment.

Pel que fa al software del C3SatP i de la MCS, des del IEEC-ICE, s'està també desenvolupant la part necessària per al control i gestió del nanosatèl·lit, tant software encastat de vol com software de terra.

Per aconseguir una millor modularitat i reutilització del codi, s'ha dissenyat i implementat un framework o software comú, que s'anomena C3SatP-Basis. El qual incorpora moltes de les funcionalitats comunes com per exemple l'enviament de missatges entre components. Com intervenen diferents plataformes amb diferents sistemes, s'ha construït aquest software de manera que la part genèrica que es igual a tots els sistemes estigui en aquest framework i les particularitats a cadascun d'ells.

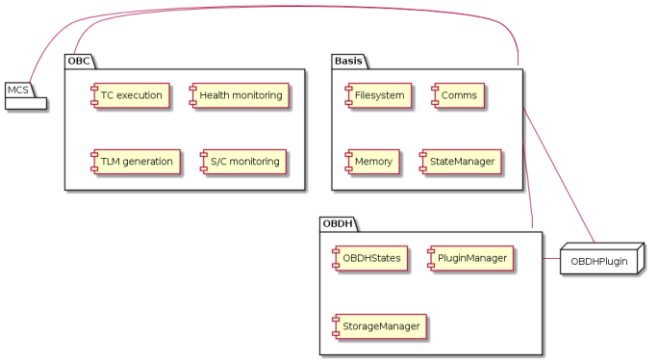


Figura 6. Diagrama de components software de tota la infraestructura relacionada amb el C3SatP

L'abast del TFG consisteix en familiaritzar-se amb el framework bàsic, incorporant alguna funcionalitat encara absent i el desenvolupament del software central del OBDH.

1.2 FINALITATS. OBJECTIUS

En relació als objectius i finalitats d'aquest TFG, hem de situar-lo en el context, ja que està emmarcat en un projecte general, i aquest projecte participa en el programa Producte presentat a la introducció d'aquest article.
La finalitat principal del programa "Producte", que pertany al programa "Industria del coneixement", es canviar el motor de l'economia del país pels propers anys [5]. Per tant, la finalitat indirecta es promoure la creació de noves empreses d'àmbit científic, així introduint un nou ecosistema al país, d'alt valor afegit, i creant nous treballs de alta qualificació.
El programa consisteix en ajudar aquestes noves empreses (spin-offs) a adquirir el grau de maduresa suficient perquè puguin accedir als mercats financers per continuar el seu desenvolupament empresarial.

Respecte a les institucions, que estan adherides al programa anteriorment mencionat, i que participen en el desenvolupament del prototip C3SatP, l'objectiu es investigar quines tecnologies son les més adients per aconseguir un prototip de CubeSat per poder ser llançat a l'espai, permetent així a l'IEEC i ICE-CSIC realitzar investigacions i col·laborar amb actuals i noves empreses que puguin sorgir.

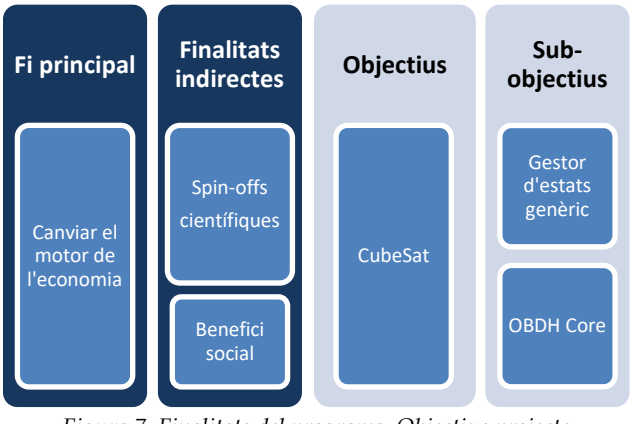


Figura 7. Finalitats del programa. Objectius projecte. Subobjectius TFG.

Pel que fa als objectius concrets d'aquest TFG, s'ha utilitzat un formalisme, que consisteix en identificar cada requisit amb un codi únic el que es pretén es aportar més claredat i una millor traçabilitat dels mateixos durant tot el procés de l'enginyeria.
Estan identificats amb la següent nomenclatura TFG-OBJ. Posteriorment hi ha una diferenciació d'objectius sent un objectiu genèric propi d'un TFG (TFG-OBJ-GEN-XXX), i d'altre banda els objectius particulars del projecte C3SatP (TFG-OBJ-CSA-XXX).

L'objectiu principal (TFG-OBJ-GEN-001) del TFG és realitzar el desenvolupament del software de control del sistema C3SaTP, mitjançant un procés d'enginyeria, el qual consisteix en fer, l'anàlisi, disseny, desenvolupament i test.

Els subobjectius que formen l'objectiu principal, estan caracteritzats per ser objectius propis de l'enginyeria del software:

- TFG-OBJ-GEN-001: Analitzar l'estat del projecte.
- TFG-OBJ-GEN-002: Fer una planificació de les tasques que s'han de desenvolupar.
- TFG-OBJ-GEN-003: Realitzar reunions amb el tutor per extreure i documentar els requisits.
- TFG-OBJ-GEN-004: Realitzar el disseny del software tenint en compte els requeriments i aplicar patrons de disseny quan sigui necessari.
- TFG-OBJ-GEN-005: Validar que es compleixen els requisits documentats i acordats amb els casos de test associats.

Específicament d'aquest projecte, tenim els següents objectius:

- TFG-OBJ-CSA-001: Entendre i analitzar la documentació i el codi proporcionat.
- TFG-OBJ-CSA-002: Analitzar, dissenyar i implementar un gestor d'estats genèric per l'OBC i l'OBDH.
- TFG-OBJ-CSA-003: Analitzar, dissenyar i implementar una arquitectura de plugin per comunicar l'OBDH amb els payloads.
- TFG-OBJ-CSA-004: Analitzar, dissenyar i implementar el software base que utilitzarà l'OBDH.

1.3 Organització del document

Aquest document esta format per la ja presentada introducció, on es detalla l'estat de l'art del projecte, les finalitats i objectius, i aquesta mateixa secció on es defineix com esta format el document. Tot seguit, es mostra la metodologia seguida per aconseguir els objectius, les fases del projecte. Com a quarta secció es mostren els resultats obtinguts a cadascuna de les fases. I per últim i cinquè punt, les conclusions que extretes a partir dels resultats i les línies futures que es poden seguir amb aquest projecte.

3 METODOLOGIA

3.1 Reunions

Degut al tipus de projecte, la metodologia seguida ha sigut estrictament fixada. Es a dir, aquest tipus de projectes segueixen una metodologia de desenvolupament de software clàssica, seguint el següent ordre: Anàlisi, Disseny, Implementació i Test. En les següents seccions es detallaran cadascuna de les fases.

Tot i així, s'ha seguit una metodologia Agile, en el sentit que s'han fet reunions freqüents, entre 1 i 2 setmanes cadascuna, a més del constant feedback amb el tutor.

En aquestes reunions es tractaven tots els temes relacionats amb el projecte:

- Codi proporcionat.
- Requisits funcionals i no funcionals.
- Com havien de ser els documents del projecte.

3.1.1 Actes de reunió

Com a decisió personal, s'ha documentat tot el que s'ha parlat en les diferents reunions. Un cop documentat, s'ha enviat al tutor per validar que el que s'ha documentat es correcte. Cadascun dels documents generats, estan identificats amb el codi TFG-MET-XX on XX és el numero de reunió.

3.2 Fase d'anàlisi

En aquesta fase es detallen els requisits funcionals i no funcionals més importants del TFG, així com les diferents activitats que s'havien de fer durant tot el procés.
El primer pas que es va realitzar al començament del projecte va ser una reunió inicial amb el tutor. En la qual es van establir els diferents objectius del TFG, a més de les activitats, les quals estan identificades (TFG-ACT-XXX) i subactivitats a realitzar:

- TFG-ACT-001: Realitzar el document de especificacions de software del projecte (RD-03).
  - TFG-ACT-001.1: Aprendre el llenguatge de programació de redactat de documents científics LaTeX.
  - TFG-ACT-001.2: Descarregar i instal·lar el software necessari per compilar la documentació (TexMaker)
- TFG-ACT-002: Extreure els requisits funcionals i no funcionals i documentar-los al RD-03.
  - TFG-ACT-002.1: Per cada reunió, s'ha documentat els punts parlats en la reunió i els comentaris fets pel tutor.
- TFG-ACT-003: Realitzar el disseny del software encarregat de gestionar els estats.
- TFG-ACT-004: Realitzar el disseny del software encarregat de gestionar la comunicació entre els plugins i l'OBDH.
- TFG-ACT-005: Realitzar el test del gestor d'estats
  - TFG-ACT-005.1: Familiaritzar-se amb l'entorn de Google Test Framework
- TFG-ACT-006: Realitzar el test del gestor de plugins.
- TFG-ACT-007: Realitzar la implementació del gestor d'estats.
  - TFG-ACT-007.1: Familiaritzar-se amb l'IDE
    - Build configurations, compilador i linker.
- TFG-ACT-008: Realitzar la implementació del gestor de plugins.

Les activitats i subactivitats pròpies de la fase d'anàlisi van ser: TFG-ACT-001, TFG-ACT-001.1, TFG-ACT-001.2, TFG-ACT-002, TFG-ACT-002.1, TFG-ACT-002.1.

Posteriorment, es va realitzar l'anàlisi de requisits per saber que es el que es necessitava pel projecte. A continuació, s'expliquen els requisits més importants a més de

3.2.1 Requisits. Document de requisits

Al tractar-se d'un projecte en curs, existeixen documents de requeriments RD-XX. Però aquests requisits són massa genèrics es va decidir fer un document de requisits nou, anomenat RD-03. L'abast d'aquest document es plasmar tots els requisits referent al software de vol (OBC i OBDH).

Els requisits han estat extrets dels diferents canals de comunicació amb el tutor (emails i reunions). D'altre banda, es divideixen en diferents mòduls:

- Requisits generals.
- Requisits del software C3SatP-Basis.
- Requisits OBDH.
- Requisits OBC.

A continuació, es mostra un exemple de com s'ha documentat els requisits en el document RD-03:

ID	REQ-SW-OBD-FN-0002
Title	StartUp State
Description	When OBDH is turning on, the OBDH software shall change this state to Loading state.
Priority	H
Verification	R
Parents	REQ-SW-OBD-FN-0001

ID	REQ-SW-OBD-FN-0003
Title	Loading State
Description	Once the OBDH is on loading state, the OBDH software shall: 1. Start communication with OBC 2. Check health status (TBD) 3. Load plugin configuration file 4. Go to idle state
Priority	H
Verification	R
Parents	REQ-SW-OBD-FN-0001

Com es mostra a la figura 8, els requisits tenen un identificador per tenir traçabilitat durant tot el projecte. A més, s'identifiquen els requisits pares, ja que pot ser que un requisit tingui dependència en uns altres.

3.2.1.1 Requisits funcionals

A grans trets, els requisits funcionals que s'han extret són funcionalitats que han de tenir els components. Per tant, aquest requisits tenen com abast definir una funcionalitat d'una part del software base C3SatP-Basis i funcionalitats principals del OBDH.

3.2.1.1.1 C3SatP-Basis. Gestor d'estats genèric

Com s'ha indicat anteriorment, C3SatP-Basis es un framework forca avançat. Aleshores, per tal de familiaritzar-se amb l'estil de codificació i amb les funcions comunes d'aquesta llibreria, es va decidir que es fes un gestor d'estats.
Es volia fer una gestor d'estats genèric perquè es pugui fer servir per tots els components, com per exemple OBC i OBDH. Cada component pot tenir un estat diferent i a més la implementació de cadascun dels estats pot ser diferent. Havia de permetre poder afegir un estat en temps d'execució.

A la següent figura podem veure gràficament com es transiciona entre cadascun dels estats segons el requisits

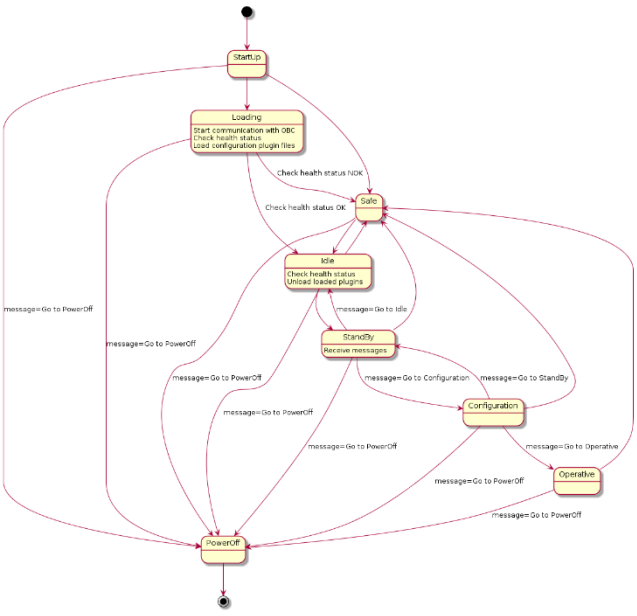


Figura 9. Diagrama d'estats del OBDH.

3.2.1.1.2 OBDH Core

Un cop dissenyada la funcionalitat del gestor d'estats es va realitzar el disseny del OBDH. Aquesta part es la més extensa del TFG ja que conté la lògica del comportament del OBDH basada en estats, gestió de plugins i gestió de dades.

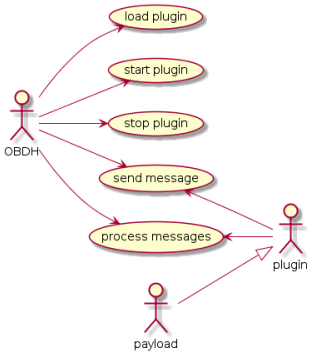
3.2.1.1.2.1 Estats del OBDH

Els estats del OBDH que seran on s'executarà tot el codi (core) són els següents: Inici, Carregant, Inactiu, Standby, Configuració, Apagat. Amb el següent diagrama (Figura 7) es pot veure les transicions entre estats:

3.2.1.1.2.2 Gestor de plugins

Per poder entendre la necessitat d'un gestor de plugins, es necessari saber que el satèl·lit esta format per dos components principals, la nau i els payloads. La nau conté l'OBC, l'OBDH, i a més antenes, panells solars, etc. Pel que fa als payloads, son components amb els quals es realitzen les mesures. Cada payload pot ser del projecte de la nau o d'un projecte extern. Per tant, no es necessari que entre ells es puguin comunicar.

Això fa que es necessites una interfície per poder comunicar els payloads amb l'OBDH per treballar conjuntament en la pressa de mesures sense comprometre el sistema en general. Per tant, s'havia de pensar en un sistema que pogués carregar cadascun dels payloads, en temps d'execució, si la salut del sistema ho permet. A més ha de poder comunicar-se amb cadascun dels payloads.
També calia tenir en compte que en aquell moment son payloads però en un futur pot ser un altre tipus de sistema, per tant ha de ser obert.



3.2.1.1.2.1 Gestor de dades

Aquesta necessitat ve determinada per la problemàtica de que el CubeSat no pot estar constantment enviant i rebre dades, ja que només podrà fer-ho quan sigui possible a nivell de recursos i de connexió amb la MSC. A més, qui s'encarrega d'executar les telecomandes, comandes enviades des de la MSC, és l'OBC, el qual té la memòria i capacitat d'emmagatzematge molt limitada. Per últim, les telecomandes no s'han d'executar en el moment en qual es reben. Per tant, el que es va decidir es utilitzar l'OBDH com intermediari per guardar les telecomandes i telemetries pendents. Aleshores, el que es volia es tenir un cua d'aquests registres.

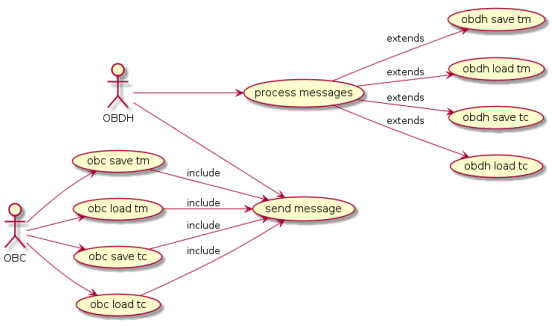


Figura 10. Diagrama de casos d'ús del gestor de dades.

3.2.1.2 Requisits no funcionals

Els requisits no funcionals més rellevants són els següents:

El llenguatge de programació serà C/C++. Es va decidir fer tot el sistema en C ja que es necessita un rendiment de la plataforma que permeti executar totes les accions en temps real. La utilització de C++ es nomes per fer els test unitaris, ja que el framework que s'utilitza esta desenvolupat en C++.

L'IDE serà Eclipse Xilinx 2017.3. Per tal de poder compilar el codi en el hardware que es farà servir al CubeSat s'utilitza la plataforma que ells han proveït.

Totes les estructures tindran definida memòria estàtica. Es a dir, un cop inicialitzades, no es podran fer reallocs.

Els fitxers de configuració seran en format XML. S'ha decidit aquest format per que es un format editable i que es pot exportar a diferents utilitats. A més proporciona la característica que es pot validar, abans de recorre'l.

Els mètodes inicialitzadors han de rebre una configuració i retornar si s'ha pogut inicialitzar o no. Han de ser idempotents, es a dir es criden un cop i després han de retornar el mateix valor, fins que es desinicialitzi.

3.3 Fase de disseny del sistema

Amb els requisits recopilats a la anterior fase, ja es podia fer el disseny de tots els mòduls sol·licitats.

3.3.1 C3SatP-Basis. Gestor d'estats genèric

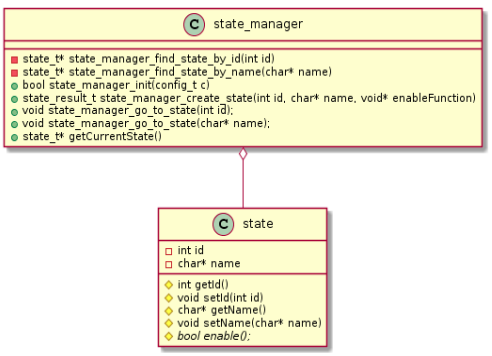


Figura 13. Gestor d'estats genèric.

Amb aquestes dues dos classes, es modela el comportament que es volia en els requisits, delegant a l'estat la decisió si s'activa o no, sense alterar el funcionament del gestor d'estats.

Per últim, permet a cada estat saber quin es l'estat actual per decidir si es valida l'activació i es compleix amb el graf de la Figura 7.
Com a patró de disseny, es una variant del State. Ja que no canvia directament l'estat del State manager però crida a un mètode seu que ho canvia.

3.3.2 Gestor de plugins

El gestor de plugins es una de les part més complexa del tfg, inicialment es va pensar com una comunicació implícita on els esdeveniment eren els comunicadors entre els plugins i les funcionalitats core. Però fent una segona reunió es va aclarir que havia de ser una comunicació explícita i no bloquejant:

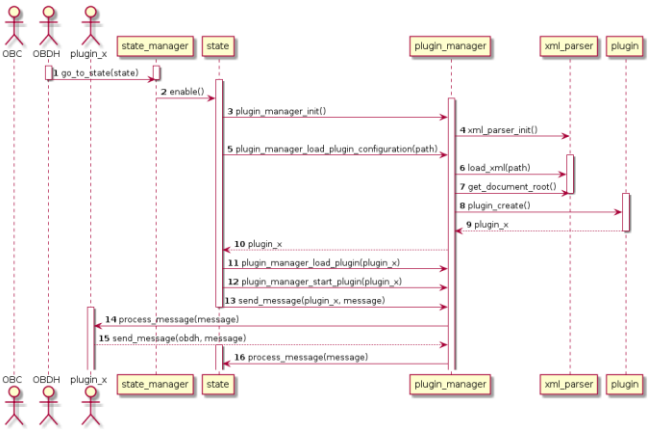


Figura 11. Diagrama de seqüencia del gestor de plugins.

Finalment, el disseny resultat fruit del anàlisi és el mostrat a continuació:

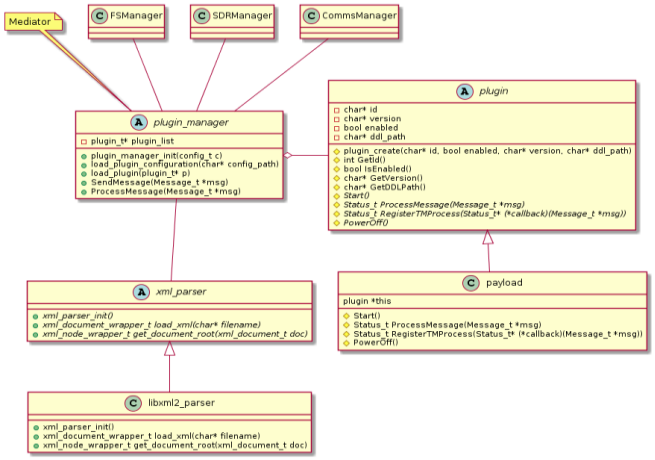


Figura 12. Diagrama de classes del gestor de plugins.

Aquest mòdul està basat en el patró de disseny de software mediator. S'han estudiat altres possibilitats de patrons de disseny que s'utilitzen en problemes semblants com pot ser el patró Observer.

La decisió d'utilitzar el patró Mediator en comptes de l'Observer és perquè la intenció de l'observer es fer una comunicació unidireccional, és a dir, l'objecte observat es el que notifica als observadors, sent tots notificats, en canvi en el patró mediator es realitza una comunicació bidireccional, ja que el mediador coneix als objectes mediats i viceversa. I per últim, en el cas de la comunicació des del mediator es una comunicació concreta, es a dir, notifica a un dels objectes referenciats.

En el cas del TFG, les parts conegudes són l'OBC i els diferents plugins que puguin existir.

D'altre banda, es necessita un parsejador xml, per carregar les configuracions de cada plugin. Per tant, hem fet una interface pública anomenada xml_parser on figuren totes les operacions comunes. Això ens ha permès poder utilitzar en aquest cas la llibreria libxml2, però en un futur si es considerat per alguna raó es pot utilitzar un altre llibreria sense afectar a la resta del software.

3.3.3 Gestor de dades

En el següent diagrama de classes es pot veure com s'ha implementat aquesta funcionalitat:

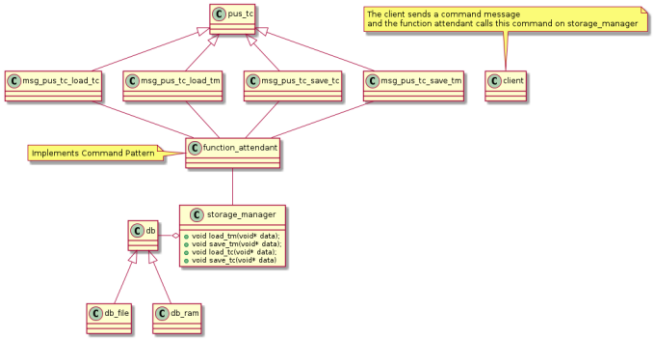


Figura 14. Diagrama de classes del gestor de dades

Com hem indicat a la fase d'anàlisi, es necessita una cua on emmagatzemar els registres. El que s'ha fet es aprofitar les funcionalitats que proporciona el framework C3SatP-Basis. En primer lloc, utilitzant el function_attendand que segueix el patró de disseny Command, permetent que cada comanda que li enviem cridi al mètode sol·licitat del storage manager. I en segon lloc, per guardar i llegir els registres s'utilitza la interfície de db, implementant en els mètodes de load i saves corresponent la forma de guardar, en termes d'ordre.

3.3 Fase de test

3.3.1 Metodologia TDD

Respecte al primer mòdul, el gestor d'estats, no s'ha pogut seguir la metodologia Test Driven Development abreujat TDD [6], esperada al principi per problemes de integració de la llibreria de Google Framework.

Es va aplicar després pel mòdul de gestor de plugins, el qual ha sigut més ràpid desenvolupar gràcies a utilitzar aquesta metodologia.

El procediment per aconseguir fer TDD era el següent:

1. Definir els headers/interface de cada mòdul, per saber quins inputs i outputs rebran cada unitat a testejar.
2. Un cop son definits, es realitza el test, mirant les particions equivalents en cada cas.

3.3.2 Test de caixa blanca

Els test que s´han realitzat són:

- Branch coverage
- Decision coverage
- Path coverage

Amb els quals, s'ha aconseguit arribar a un 80-90% del codi testejat. Tot i que hi han camins que no han pogut ser testejats per la peculiaritat que no reben paràmetres i només es donaran aquests casos extrems, com per exemple, si el sistema sense memòria no es podrà reservar més memòria.

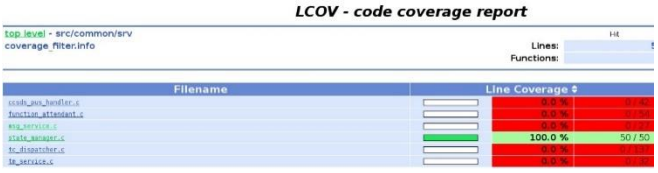


Figura 16. Cobertura de codi del state manager

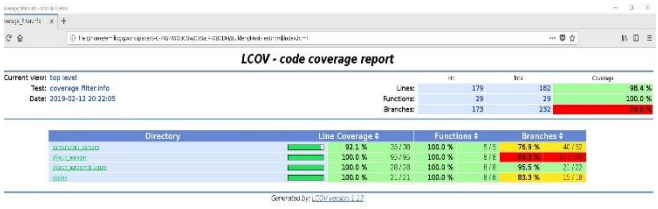


Figura 15. Cobertura de codi del mòduls de l'obdh.

3.3.3 Google Test Framework

Per testejar el software s'ha fet servir aquest API [7]. Amb la qual ens permet testejar el software creat de manera fàcil:

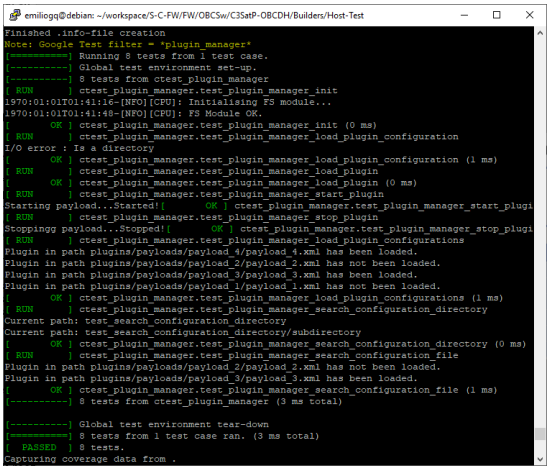


Figura 17. Execució dels test del plugin manager

La utilització d'aquest framework va comportar que un dels requisits no funcional fos utilitzar programació en C++, ja que aquesta llibreria esta feta en C++. Tot i així, el que s'ha fet per solucionar aquest problema es fer una classe en C++ que cridi a un fitxer .test programat en C que es realment on s'implementen els test.

A més, el que s'ha fet es fer un embolcall per evitar en cas de que la llibreria canviï, no s'hagi de canviar tot el codi de test. Aquest embolcall fa referencia als mètodes del framework i el codi de test crida als mètodes embolcalls.

3.4. Implementació

Aquest ha sigut la fase més complicada del projecte, degut al desconeixement de les eines utilitzades en l'IDE com les build configurations, el compilador GCC/G++, i el linker.

3.4.1 Build configurations

S'han realitzat diferents configuracions pels diferents projectes software. Això permetia poder fer amb el mateix codi, compilar per diferents plataformes o propòsits.

C3SatP-Basis: S'han utilitzat 3 tipus de configuracions, tot hi que n'hi han més:

- Debug: Compila amb GCC, el codi en C i genera una llibreria estàtica per poder utilitzar a altres projectes, com per exemple al C3SatP-OBDH.
- Debug-Gtest: Compila amb G++ els fitxers de codi C i genera una llibreria estatica, per
- Debug-test: Compila amb G++ els fitxers de codi C i genera un executable per poder ser testejat. Utilitzat posteriorment amb Google Test Framework.

C3SatP-OBDH: S'han utilitzat 2 tipus de configuració.

- Host-Debug: Compila amb GCC el codi C i genera executable per ser executat en un Linux.
- Host-Test: Compila amb G++ el codi C i genera executable per ser testejat amb Google Test.

3.4.2 Utilització de la llibreria C3SatP-Basis

Un dels motius de fer dos projectes de software al mateix temps es per aprofitar el software creat en el C3SatP-Basis per poder utilitzar-lo a la resta de projectes, com al OBDH o OBC. Per això, es necessari un software obert a diferents casuístiques.

3.4.3. Utilització de la llibreria Libxml2

S'ha utilitzat la llibreria libxml2 [8] per llegir els fitxers de configuració com per exemple els payloads, o la configuració del plugin manager. També s'ha utilitzat aquesta mateixa llibreria per validar els propis fitxers XML utilitzant fitxers DTD [9] per aquest propòsit.

3.4.4 Càrrega de llibreries en execució

Per al mòdul del gestió de plugins, hem utilitzat la llibreria de carrega dinàmica de llibreries [10]. Carreguem la llibreria del plugin i després cridem als mètodes necessaris per realitzar la comunicació.

3.4.5 Control de versions

Pel que fa al control de versions del codi, s'ha fet servir un repositori de Git. La forma de treballar ha sigut la següent:

1. Al inici del projecte es va fer una branca pública/remota "tfg-emiliogq"
2. Posteriorment es va treballar localment en les diferents funcionalitats, fent per cadascuna una branca local.
3. Quan es finalitzava el mòdul, es feia la fusió amb la branca publica i es pujava al servidor remot.

4 RESULTATS

Arribat fins a aquest punt, on és el més important del projecte, ja que es on es veu si els objectius s'han completat.
Els mètodes d'avaluació dels resultats son varis, però el més simple es verificar si cadascun dels requisits funcionals que s'han extret durant el projecte es compleixen o no.

Com són més de 30 requisits, a continuació es mostrarà nomes el grau de compliment dels objectius marcats al inici del projecte:

Objectiu	Completat
TFG-OBJ-CSA-001	100%
TFG-OBJ-CSA-002	100%
TFG-OBJ-CSA-003	90%
TFG-OBJ-CSA-004	50%

Els objectius no completats com TFG-OBJ-CSA-003 i TFG-OBJ-CSA-004, son deguts a que la fase de implementació no està finalitzada.

Tot i no haver completat tots els objectius, s'ha de tenir en compte també tots els documents generats a cada una de les fases:

- Document d'especificació de requisits (RD-03)
- Diagrames de disseny
- Codi testejat
- Documentació del codi.

5 Conclusió

Com es pot comprovar objectivament al apartat anterior, no s'han completat tots els objectius. Això ha sigut degut als següents factors:

- Ús avançat del llenguatge a desenvolupar. En l'àmbit acadèmic, s'expliquen nocions molt bàsiques del llenguatge C, que en l'entorn empresarial no son suficients. És per això, que calen moltes més hores.
- Desconeixement de les eines utilitzades en aquest projecte. Com s'ha comentat al llarg del article, s'ha tingut molts problemes per compilar les llibreries, tant externes com internes. En l'àmbit acadèmic ens ensenyen el llenguatge C, en entorn Windows on l'IDE Visual Studio et deixa transparent tota aquesta feina.
- Situació personal. El fet de haver de fer Erasmus, i aconseguir diners per poder permetre-ho econòmicament, ha implicat que hagi de treballar de dilluns a diumenge, reduint així la meva disponibilitat.

D'Altre banda, s'ha vist com s'implementa un projecte software en l'ambit del espai amb les directrius que marca l'enginyeria del software.

5.1 Línies futures

Aquest TFG es un projecte de software que està a dins d'un projecte més gran, per tant, l'abast d'aquest es pot dimensionar tant com el projecte pare és.

Una de les línies que es pot aconseguir es fer que cada plugin estigui en un contenidor Docker, per tal de aïllar el sistema de possibles caigudes d'algun dels plugins.

També es pot investigar sobre les diferents llibreries de monitorització del sistema, cpu, disc, i memòria. En aquest treball, s'ha trobat lm-sensor, pero no s'ha pogut investigar gaire per falta de temps.

Agraïments

Agraeixo al meu tutor, Lluis Gesa, que ha donat moltes facilitats per establir comunicació en qualsevol moment que sorgien dubtes.

A la meva parella, per estar al meu costat quan més ho necessitat.

Bibliografia

[1] Erik Kulu, Nanosat, https://www.nanosats.eu/cubesat.html
[2] Interorbital Systems, June 2018. "News", http://www.interorbital.com/News
[3] Project Calliope. Oct, 2011. "Cubesat eating a Tubesat", https://www.science20.com/satellite_diaries/cubesat_eating_tubesat-83273
[4] California Institute of Technology, Mars Cube One (MarCO). Mission Overview.
[5] Generalitat de Catalunya, May, 2018. "Programa industría del coneixement", http://agaur.gencat.cat/ca/beques-i-ajuts/pagines-especials/recerca/programa-industria-del-coneixement/
[6] Wikipedia, "TDD", https://es.wikipedia.org/wiki/Desarrollo_guiado_por_pruebas
[7] Google, "Google Test Framework", https://github.com/google/googletest
[8] LibXml2, "Xml Examples", http://www.xmlsoft.org/examples/
[9] W3Schools, "DTD Introduction", https://www.w3schools.com/xml/xml_dtd_intro.asp
[10] "Dynamic libraries", https://dwheeler.com/program-library/Program-Library-HOWTO/x172.html
[11] IEEE, "Writing your abstract", https://journals.ieeeauthorcenter.ieee.org/create-your-ieee-article/create-the-text-of-your-article/writing-your-abstract/
[12] Gestio de projectes, Des, 2009. "Com fer un article"
[13] Gesa Bote, L. Gen, 2018. C3SatP-DD-01: Product Requirements Document. Cerdanyola del Vallés.
[14] Gesa Bote, L. Set, 2018; C3SatP-DD-03: Software Design Document.
[15] Gesa Bote, L. Jul, 2018; C3SatP-RD-01: Product Definition Document.
[16] Richards, M. Agost. 2015; *Microkernel architecture*. https://www.oreilly.com/ideas/software-architecture-patterns/page/4/microkernel-architecture