Aalto University
School of Electrical
Engineering

Combining rapid development and ECSS standards: on-board computer development for the ICEYE mission

23 September 2014
Tartu Conference on Space Science and Technology
Hannu Leppinen

Contents

- ICEYE project intro
- Development practices
- OBC development
- Results and status
- Summary

Aalto university main building

Arktika jääväljad Foto: Timo Palo

Soome firma lubab saata kosmosesse kuus satelliiti

Avaldatud: 13. mai Kell 17:11

Soome esimene satelliidifirma hakkab müüma täpset teavet Arktika jääolude kohta, lootes sellega teenida sadu milioneid eurosid, vahendab ERR uudisteportaal

Kauppalehti

TIISTAINA
13. TOUKOKUUTA 2014

WWW.KAUPPALEHTI.FI / MOBIILI: M.KL.FI / APP.KAUPPALEHTI.FI
PERUSTETTU VUONNA 1898

Matkalla avaruuteen

Antti Kestilän, Pekka Laurilan ja Rafael Modrzewskin yritys on ensimmäisenä suomalaisena lähdössä kaupallisille satelliittimarkkinoille. Arktista jääinformaatiota tarjoava yritys tähtää satojen miljoonien eurojen liikevaihtoon. OMA YRITYS » 16–17

A UUTISET

Amerikkalaisten kulutuskysyntä

USA:n talous voi yllättää iloisesti » 6–7

B REPORTAASI

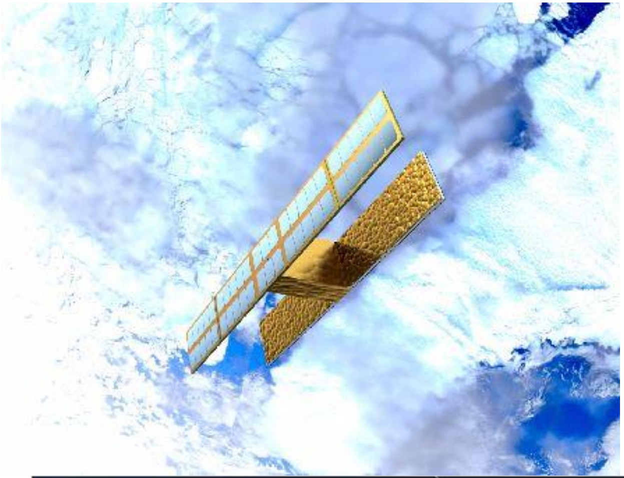
Japani käynnistää varovasti ydinvoimaloitaan » 12–15

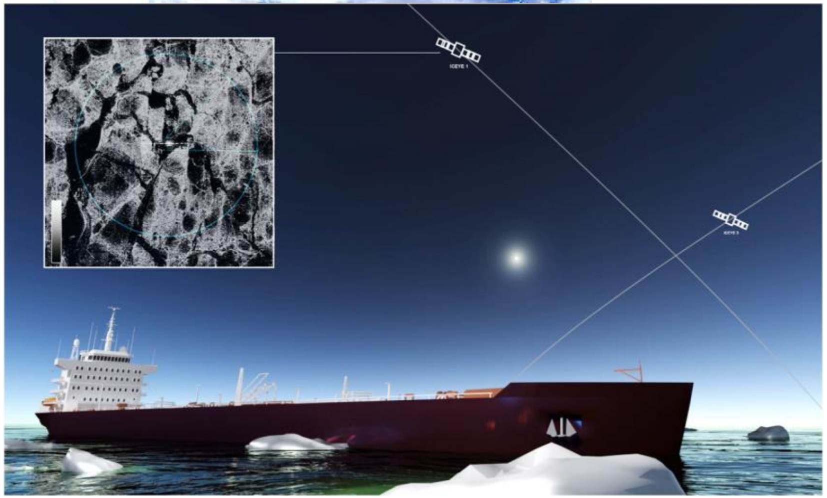
C PÖRSSI

Netflixiä naitetaan Applelle » 19

ICEYE project

- Shipping traffic and oil exploration in the Arctic are increasing
- Ice information vital to safe operations
- Aalto startup idea: constellation of low-cost SAR microsatellites (50 kg) to provide commercial, high refresh rate ice information





Current project status

- **Customers identified, negotiations ongoing**
- **Satellite design and subsystems mostly frozen, EM manufacturing and procurement started**
- **SAR instrument currently being tested with a flight campaign**
- **Transitioning from university project to a private company**
- **Demo satellite 2016**

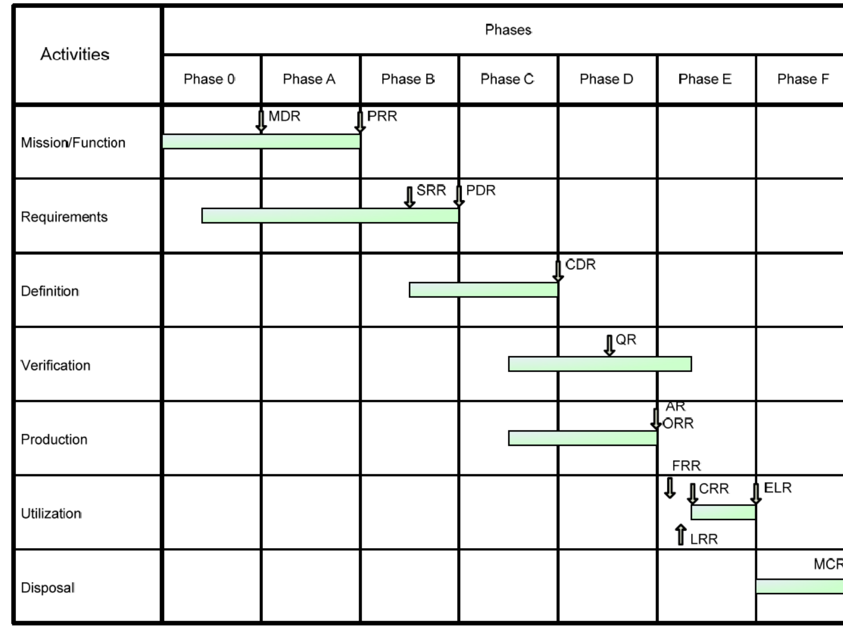
Aalto small satellite development process

- **Problems with university space projects:**
  - High quality engineering needed
  - Students may lack time and skill

- **Answer: learn both from space industry best practices and the agile community**
  - Extract maximum contribution from each student
  - If necessary, outsource the most difficult parts
  - Used in Aalto-1, Aalto-2

How is ECSS related?

- **European Cooperation for Space Standardization**
- **“necessary evil” when co-operating with existing space industry**
- **pedantic following requires mountains of documentation**
- **however: valuable lessons can be extracted**



The notorious waterfall model

ICEYE approach

- **Based on Aalto university small satellite development practices**
  - Workforce mostly recruited from talented/distinguished Aalto-1 and Aalto-2 contributors
- **Many subsystems outsourced to reduce cost and development time**
  - Knowing ECSS becomes a necessity
  - For example: commercial ground stations and mission control software require ECSS-E-70-41A compatibility

Combining ECSS and ICEYE methods

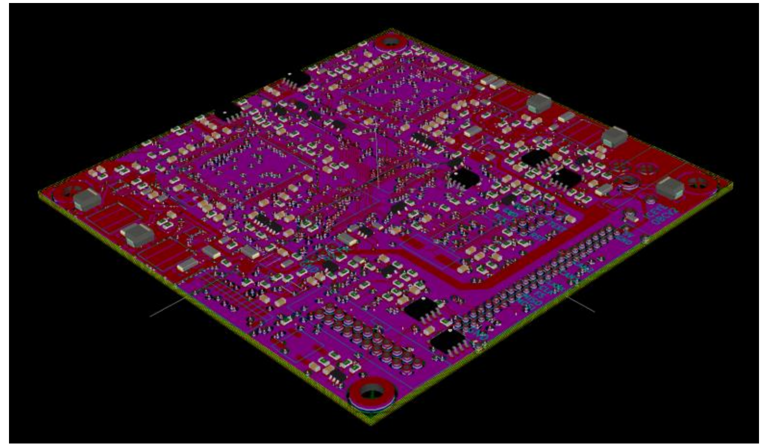
- **The spacecraft system tree starts at the satellite level**
  - Satellite-level documentation written down; rest of the documentation is in the design
- **Formal ECSS type readiness reviews (PDR, CDR, etc) discarded or heavily downsized in favour of weekly scrum-like meetings**
  - However: the purpose of these reviews is not neglected
- **Open office: everyone works in the same room**
  - 15 people in 50 m^2: information is communicated instantly when needed

Open office (half of it)

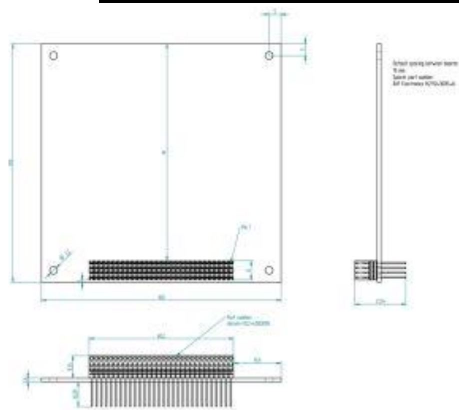
ICEYE design choices

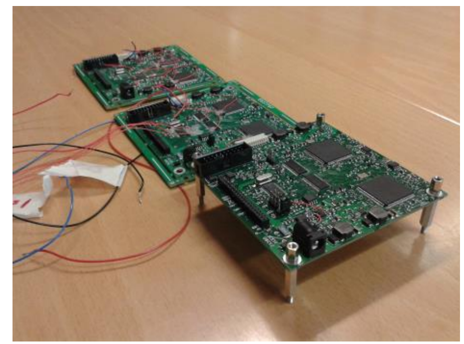
- **COTS (not a big surprise ☺)**
- **Use CubeSat solutions in microsatellite form factor**
  - Many outsourced subsystems have CubeSat heritage
- **Do extensive research to find component lists with flight heritage – minimize need for in-house testing**
- **Design life 2 years at 400 km – radiation hazards comparatively mild**
  - That's why ISS is there
  - Latchups the biggest concern → include overcurrent protection to all electronics

Hardware prototyping



- **Prototypes constantly built and made available in the open office**
  - Form factors and mechanical interfaces visible at a glance
- **Interfacing to other systems is not a part of an “integration phase” but a part of early development**

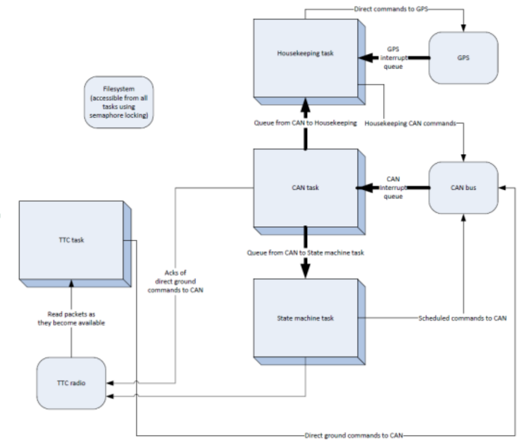




Software engineering

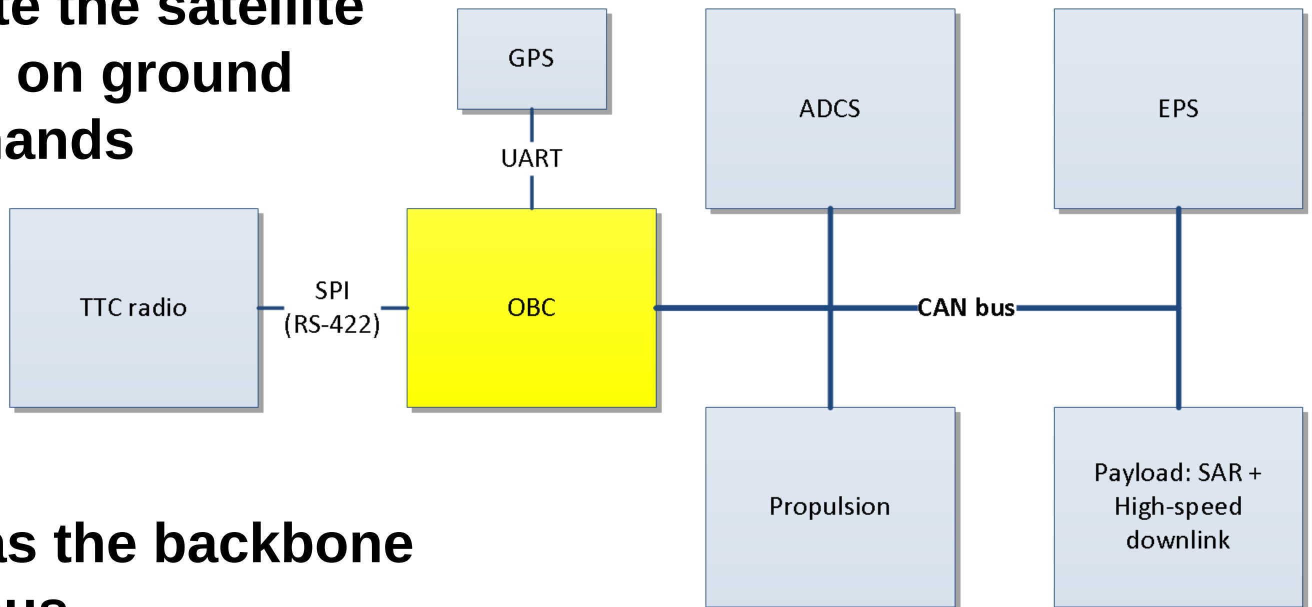
- **Texas Instruments dev tools → automotive industry best practices**
- **GitHub**
- **FreeRTOS**
- **Self-documenting code**
  - Less paper documentation to write
- **ECSS-E-70-41A**

```
// Initialize the uplink (and make sure there isn't any currently)
memset(&transfer, 0, sizeof(Transfer)); // Allocate transfer information to 0s

// Initialize the transfer.
if (init_uplink(size, filePath)){
    DEBUG("ERROR: Transfer could not be initialized.\r\n");
    return FAILURE;
}

// Start receiving packets (see analogy with ttcHandlerTask infinite loop)

EcssPacket tcPacket = {0}; // For incoming packets to be read
int retval; // Just the help variable to store the return value

// Start the timer
if (startTimer()) {
    DEBUG("Exiting.");
    return FAILURE;
}

// Send uplink reception for seq number 0 to get new transfers
ul_rec_ack(0);

/*
 * Start the actual large data transfer loop
 * */

while (datatransfer_flag == 0) { // Continue the loop as long as this flag is not set (will be set after the end of transfer)

    memset(packet_buffer, 0, ECSS_MAX_PACKET_SIZE); // Reset buffer

    int success = read_packet_from_ttc(&tcPacket);

    const EcssPacket* const pak = &tcPacket; // Copy the read packet to const value, which is used in the loop

    if(success == 0){

        // Accept only those commands which are for large data transfer

        switch (pak->srv_id){

            case ECSS_SRV_DATA_UP_PART_FIRST:
                DEBUG("Received the first packet.\r\n");
                retval = ul_first(pak);

                if (retval == FAILURE || retval == -1){
                    DEBUG("Error occured. Ending.\r\n");
                    datatransfer_flag = 1;
                }
                if (retval == END_TRANSFER){
                    DEBUG("The only packet received. Ending.\r\n");
                    datatransfer_flag = 1;
                }

                break;

            case ECSS_SRV_DATA_UP_PART_INTER:
                DEBUG("Received the intermediate packet.\r\n");
```



ICEYE system diagram

- **OBC main function: operate the satellite based on ground commands**



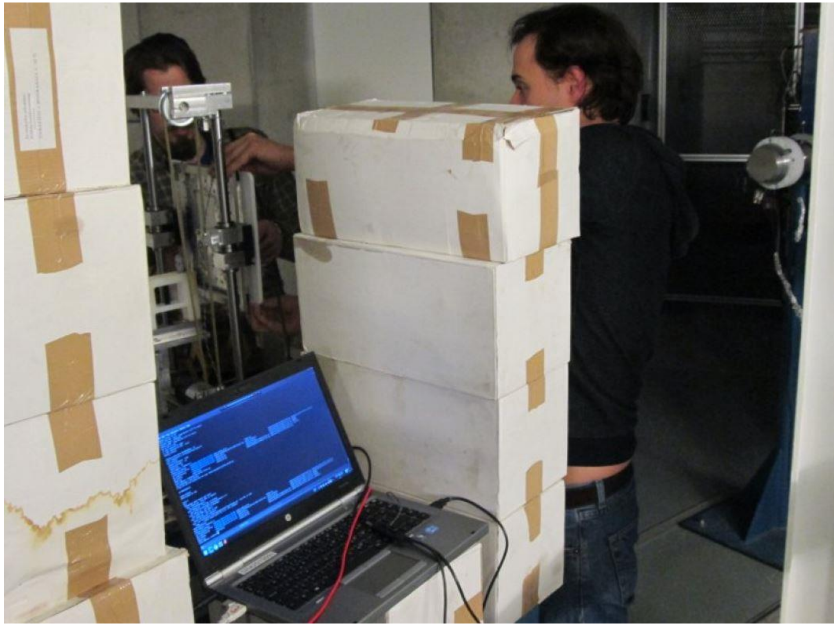
- **CAN as the backbone data bus**

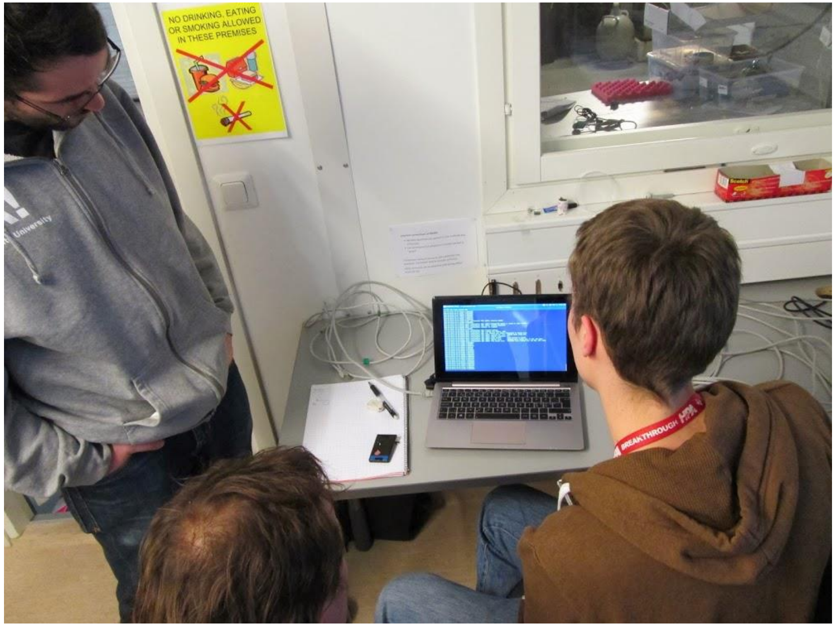
OBC development

- **HW and SW developed in parallel**
  - SW development started with development boards, but always moves to new version of hardware as it becomes available
- **Dev boards used to simulate other systems in the CAN bus**
- **Qualification tests started with available protos**

OBC radiation qualification

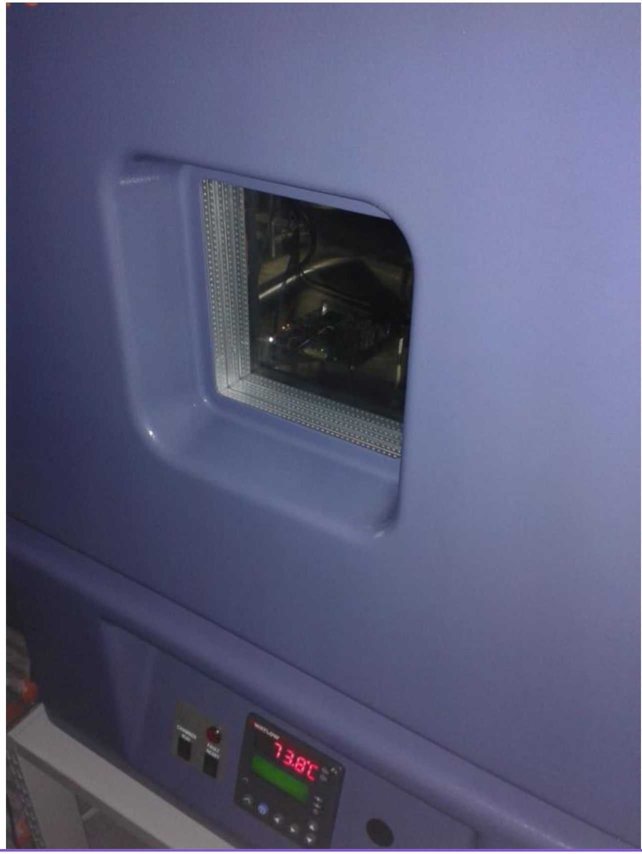
- **Aalto-1 and ICEYE devices irradiated at Jyväskylä RADEF**
- **60 MeV protons up to 25 krad dose – combined SEE & TID**





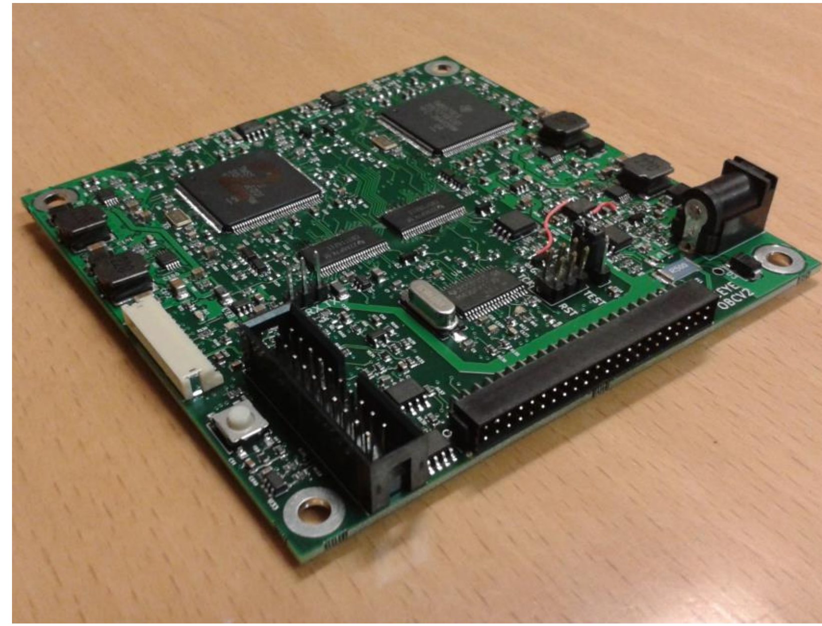
OBC thermal qualification

- Several iterations of -40C to +100C cycling
- No permanent changes - all anomalies software-related



Current OBC status

- **OBC HW v2 qualified**
- **SW framework ready; requires interface definitions from other less ready subsystems**
- **OBC HW v3 will be the Qualification Model**
  - Change to flight connectors, otherwise identical



Lessons learned

- **Set high-level requirements BUT allow flexibility for developers**
- **Monitor progress of each development branch at least weekly and reassign work force flexibly**
- **Build physical prototypes constantly**
  - Both fun and useful
- **Encourage everyone to constantly expand their areas of expertise**
- **Remember: other organizations may not work as fast as you – start procurements etc. early**
  - Our university bureaucracy has been a good teacher

Conclusions

- ICEYE constellation for commercial ice information
- ICEYE development process aims to extract maximum contribution from each team member
- ECSS is studied but not followed pedantically
- Aiming for ECSS compatibility influenced OBC design

Thank you!