Applying Shlaer-Mellor OO technology for a small embedded system

SMUG 2000

Yukitoshi Okumura, Takao Futagami

TOYO Corporation

Contents

1. Project profile
2. Challenges and Solutions
3. Results and Metrics
4. Summary

1. Project profile

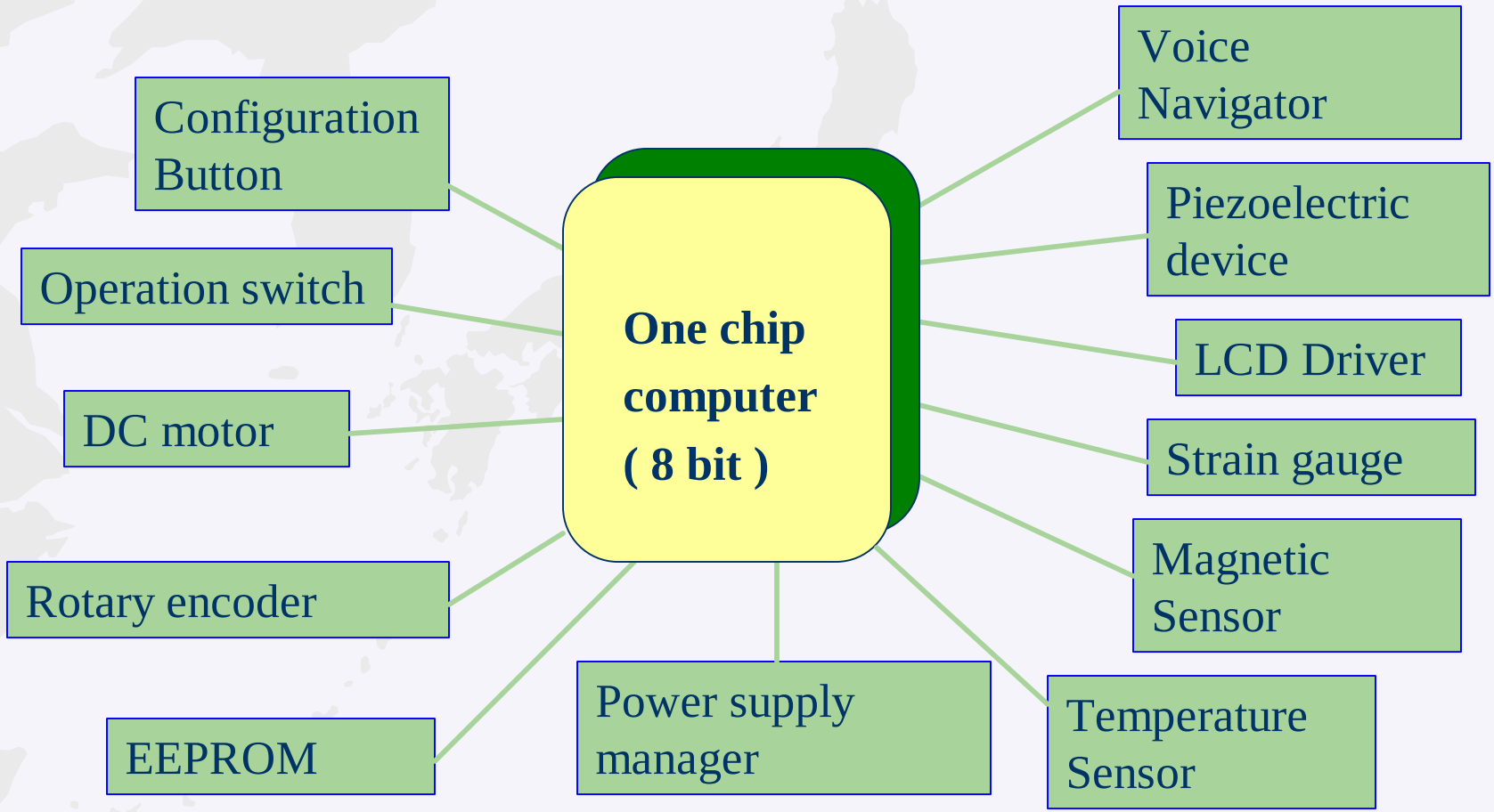
- ***Software development for medical product control***
  - ***high reliability and quality***
- ***Restricted hardware environment***
  - ***8 bit CPU(60KB ROM, 1.7KB RAM)***
    - ***usually non OO technology target***
    - ***non automatic code generation***
  - ***However, huge market in Japan***





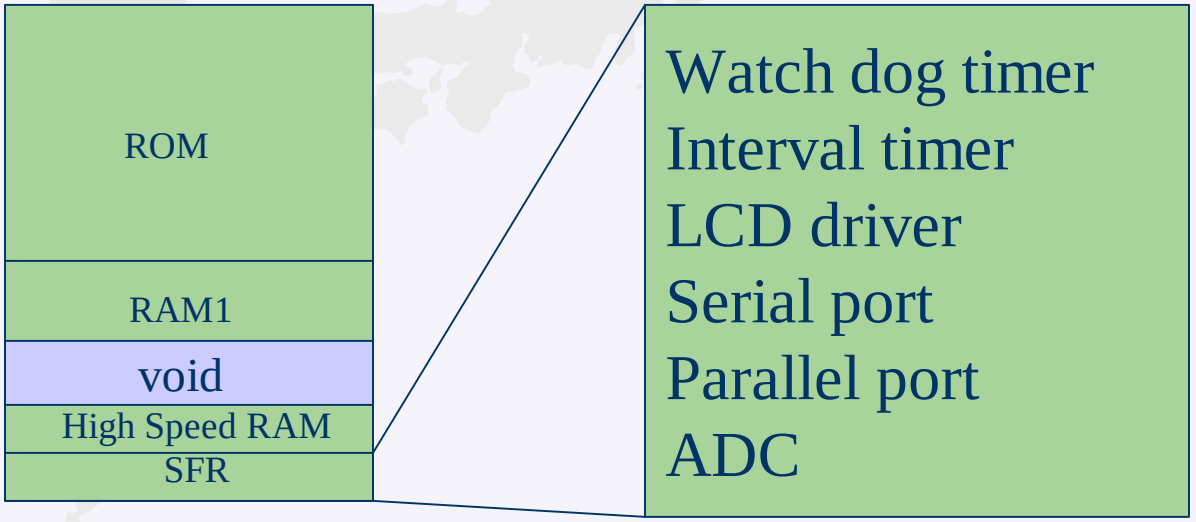
Hardware Configuration

- **Compact but typical embedded system**



Memory map

- 8 bit general purpose computer
- Divided RAMs
  - RAM1 : 1KB, High speed RAM : 0.7KB

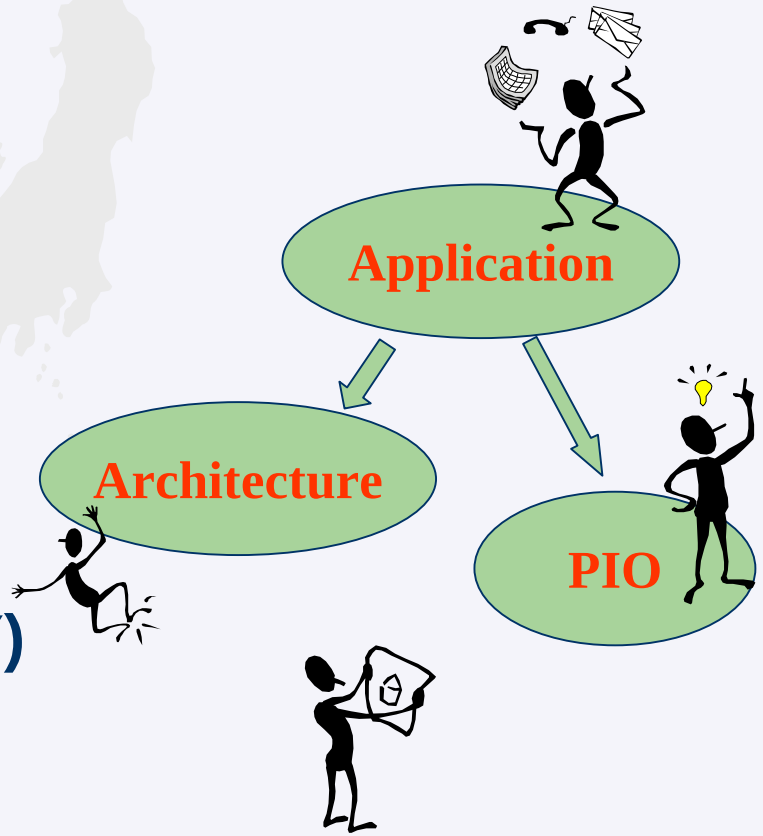


Developing Environment

- C cross compiler, CPU specific Assembler
  - NOT C++, C language is mandatory for the 8 bit world!
- ICE
- Source code debugger(but limited functions)
  - NO printf, NO casting, NO structure inspection
- BridgePoint 4.2
- DesignPoint MC-3020(enhanced)
  - Off-the-shelf C architecture
  - URL : http://www.projtech.com

Project team : 3 persons

- Requirement analyst(T)
  - interview and feasibility
- Three domain experts
  - *Application* : application analyst(M)
  - *PIO* : device driver specialist(T)
  - *Architecture* : architect(Y)



2. Challenges and Solutions

- Project challenges;
  - Limited memory
  - High performance
  - Hard realtime/Interruption
- Solutions
  - Reasonable OOA Modeling
  - Proper hand-coding introduction
  - Architecture enhancement



MC3020 enhancements

- Off-the-shelf architecture MC3020
  - ANSI-C, non RTOS, single task
  - compact code for 8, 16, 32 bit CPU
  - static memory allocation based on arrays
- MC3020 enhancement
  - more memory saving and performance
  - Easy integration with hand-coding
  - Most Japanese 8 bit compilers are not ANSI-C compliant
  - Direct source code generation from Japanese OOA model

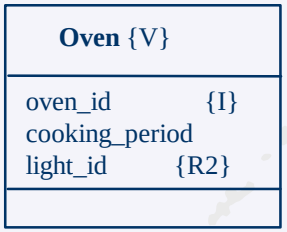
Static instance population

- Generated code optimization(less memory)
  - no create/delete functions
  - read only class : minimize RAM usage
  - no initial code : short boot time
- Easy Maintainability without model modification

Instance table

oven_id	cooking_period	light_id	current_state
1	10	1	1
2	20	2	1

Analysis model

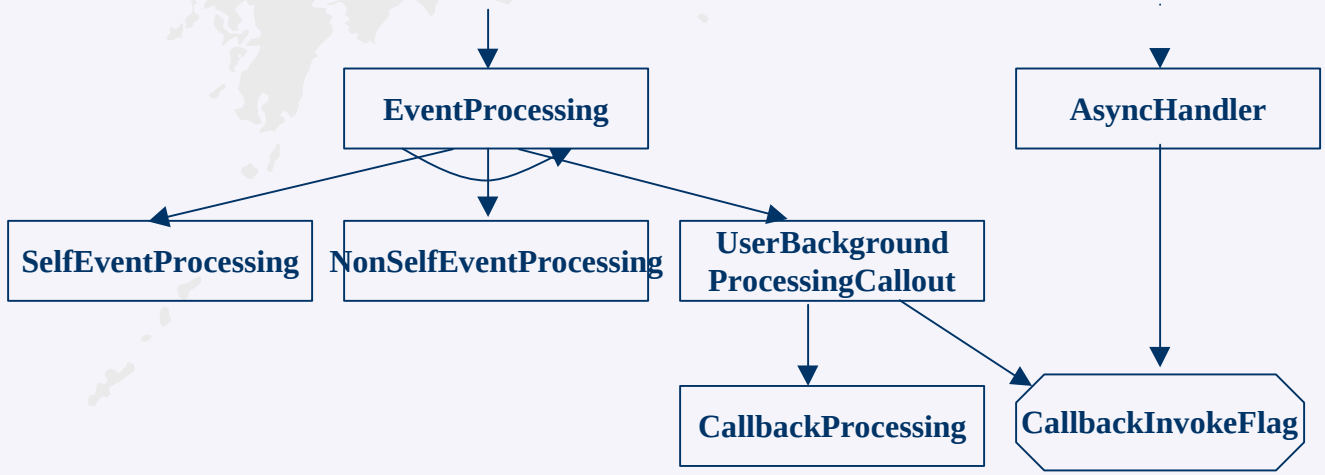


Generated code

```
O_V_s O_V_ObjectPool_s[ O_V_MAX_EXTENT_SIZE ] =
{
  /* oven_id, cooking_period, mc_L_R2, current_state */
  {1, 10, &O_L_ObjectPool_s[0], O_V_STATE_1},
  {2, 10, &O_L_ObjectPool_s[1], O_V_STATE_1}
};
```

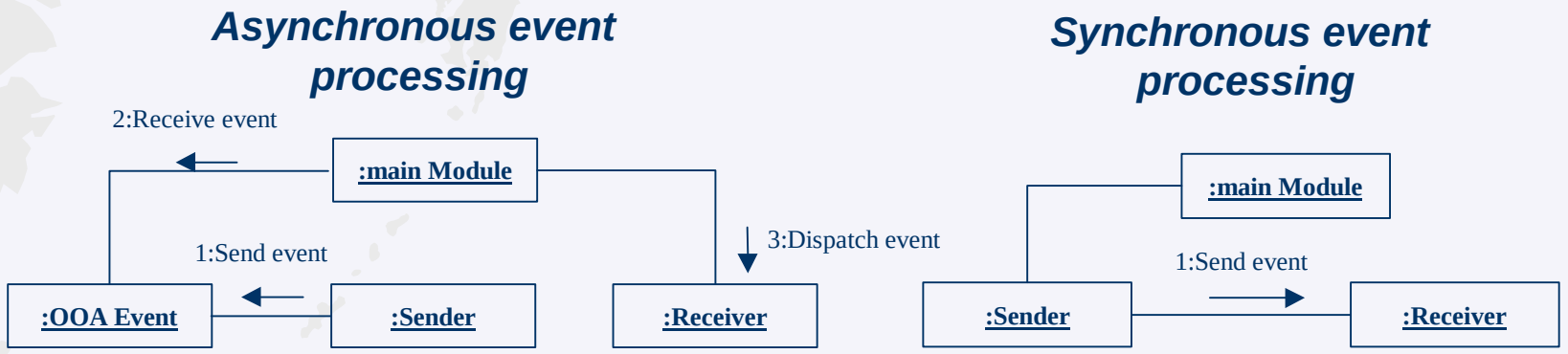
Asynchronous Service support

- Hard realtime portion
  - Code generation of interrupt
- Soft realtime portion
  - Define callback function with asynchronous safe
  - Easy and safe interface to OOA task



Synchronized event processing

- High performance and memory saving
- Async/Sync can be selected by coloring : non model modification
- But, careful modeling is necessary!

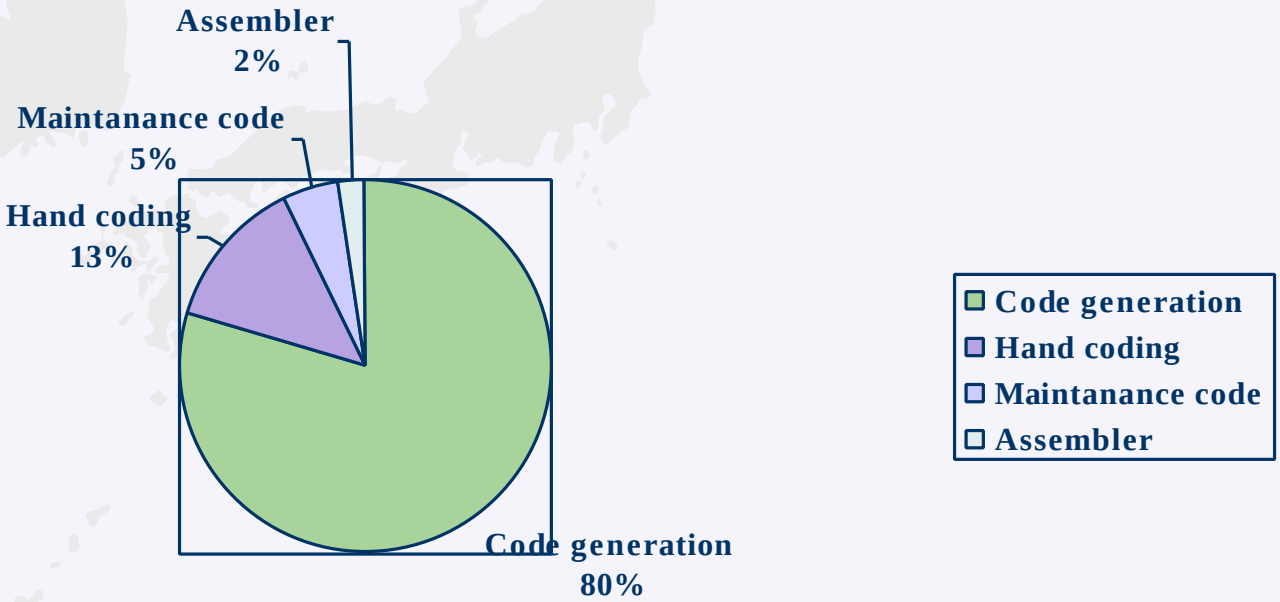


3. Results and Metrics

- Development finished, then...
  - Software size ?
  - Productivity ?
  - Code Quality ?
  - Maintainability and reuse ?

Software size

- Luckily, ROM/RAM remaining... !
- Total lines of code : 24 KLOC



Man efforts for developing

- **Total man efforts: 5.2 MM**
  - **Requirement Analysis (50 pages) : 12 days**
  - **OOA(30 Objects, 19 KLOC) : 18 days**
  - **Architecture porting/enhancement : 12 days**
  - **Software configuration : 5 days**
  - **Hand coding(Driver, 5 KLOC) : 57 days**

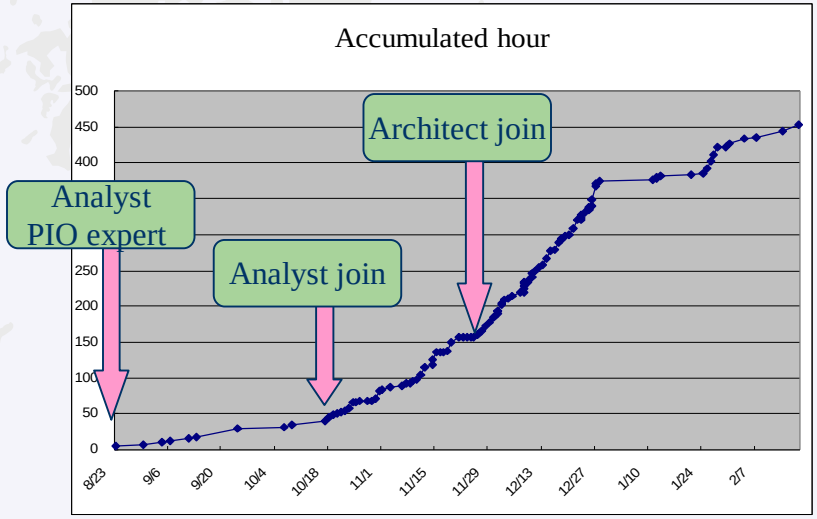
Productivity Metrics

- **OOA Productivity : 1.7 day/object**
- **C Source code productivity : 4600 LOC/MM**



Why so high productivity ?

- Shlaer-Mellor OO developing process
- Parallel developing
  - Use off-the-shelf architecture
  - Three distinct domain experts, especially architecture has been developed separately



Code Quality Metrics

- Quality is measured by a commercial static code analyzer QAC
  - e.g.. uninitialized value, illegal cast, etc.
  - URL: http://www.programmingreseach.com
- Measured Metrics of generated code
  - warning message : 1.7 warn/KLOC
  - MaCabe cyclomatic complexity : 4 functions(3%) over 20

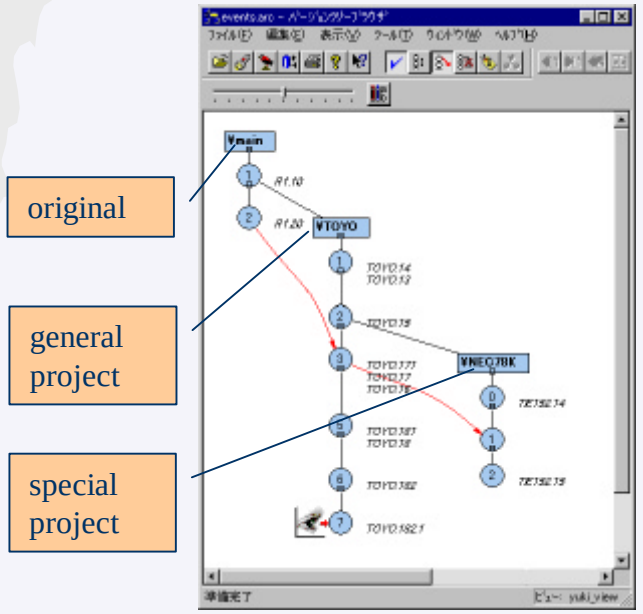


Maintainability and Reusable

- OOA model maintenance
  - controlled a number of requirement changes during development
  - minimum maintenance : just instance data modification !
- Architecture domain reuse
  - original general architecture is 100 % reusable

Architecture configuration management and reuse

- Parallel development by a commercial CM tool (ClearCase)
- Separate user specific archetype to minimize customization
  - machine dependent code
  - compiler dependent code
- Architectures for original and general project are 100 % reusable



4. Summary

- Shlaer-Mellor OO is also powerful developing technology in small Japanese embedded systems
  - quality
  - productivity
  - reuse
- However,
  - few experts for this technology
  - then,

Future

- Need more embedded OO engineers
  - more domain experts
- Need more modeling know-how
  - good domain partitioning example
  - We are waiting for more publications!
    - We don't need just UML notation book any more!
- More OOA reuse
  - more OOA modeling patterns

