Zeitschrift: Bulletin Electrosuisse
Herausgeber: Electrosuisse, Verband für Elektro-, Energie- und Informationstechnik
Band: 101 (2010)
Heft: (10)

Artikel: Le logiciel embarqué
Autor: Rieder, Medard / Gabioud, Dominique / Steiner, Rico
DOI: https://doi.org/10.5169/seals-856142

Nutzungsbedingungen
Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Mehr erfahren

Conditions d'utilisation
L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. En savoir plus

Terms of use
The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. Find out more

Download PDF: 12.07.2025

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

Le logiciel embarqué

L'importance de la formation et de l'ingénierie pour le développement du logiciel embarqué

Les systèmes embarqués – et notamment les processeurs qui sont à leur cœur – ont connu une évolution exponentielle de leurs performances. Or, la complexité croissante du matériel induit une complexité équivalente du développement du logiciel embarqué. Dans ce contexte, l'ingénierie du logiciel représente un challenge pour les PME qui développent des systèmes embarqués. Cet article se propose d'éclairer cette problématique et de donner quelques pistes à suivre.

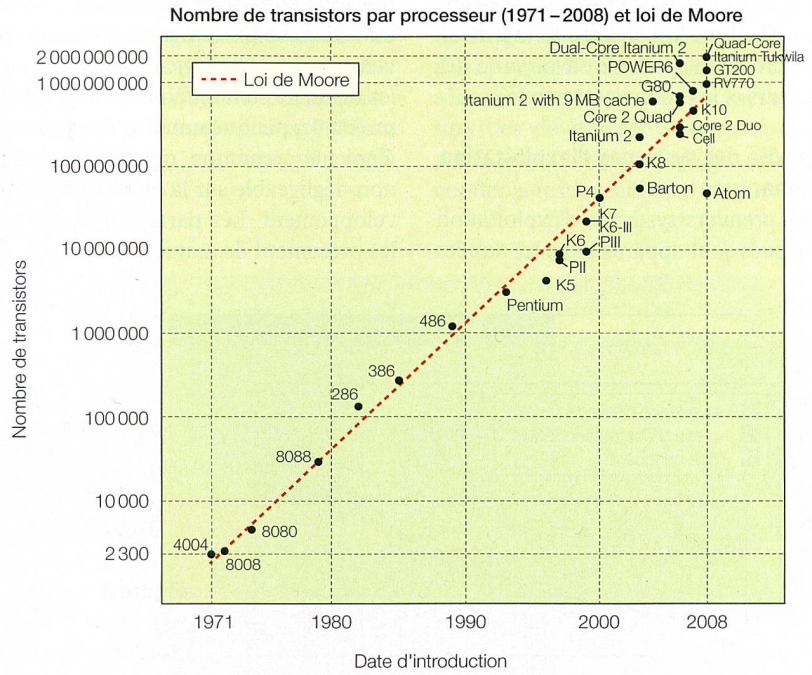
Medard Rieder, Dominique Gabioud et Rico Steiner

Tout le monde connaît l'omniprésence des ordinateurs dans notre vie quotidienne. Par contre, il est peut-être moins évident pour tous de se rendre compte que plus de 90 % des microprocesseurs se trouvent (sont « embarqués ») dans des appareils aussi banals qu'une carte de crédit ou un robot ménager.

Les microprocesseurs embarqués représentent donc un immense marché. Aujourd'hui, environ 15 milliards de processeurs sont vendus par année pour un chiffre d'affaire de l'ordre de grandeur de 100 mia. €. Un chiffre en constante augmentation, certains prédisent même que 40 milliards de processeurs seront vendus par année aux alentours de 2020.

Historique

Au cours du dernier demi-siècle, l'informatique n'a cessé de progresser, que ce soit du point de vue des ordinateurs, des systèmes embarqués ou du développement du logiciel embarqué.



Wikipedia [1]

Figure 1 L'évolution des processeurs Intel.
Le nombre de transistors les constituant a doublé tous les 2 ans, conformément à la loi de Moore représentée par la droite.

L'évolution des ordinateurs

Si les années 70 ont été les années des ordinateurs centraux, partagés par un grand nombre d'utilisateurs, les années 80 ont vu un changement de paradigme : les ordinateurs sont devenus personnels, avant de former des réseaux pendant les années 90. Puis, au début des années 2000, les ordinateurs mobiles ont pris leur essor. Et nous n'en resterons pas là, puisque dans les années 2020, les utilisateurs seront entourés par des ordinateurs plus ou moins visibles, mais omniprésents.

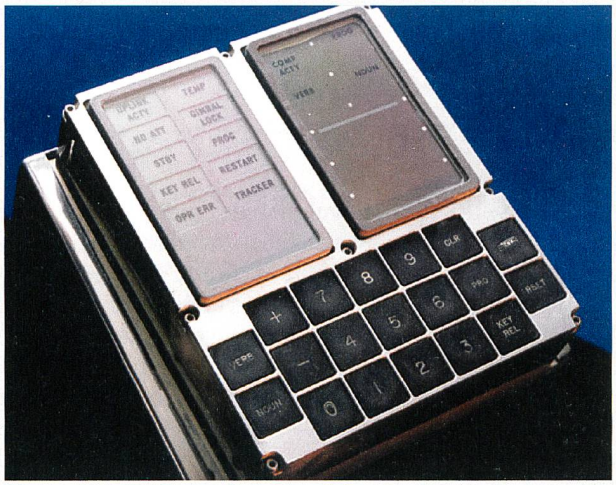
Cette évolution a été rendue possible car la densité d'intégration des circuits a progressé au rythme prédit par Moore, c'est-à-dire qu'elle a doublé tous les 2 ans (voir **figure 1** pour les processeurs de la marque Intel).

L'évolution des systèmes embarqués

Un des premiers systèmes embarqués à base de circuits intégrés a été l'AGC (Apollo Guidance Computer) utilisé pour les vols à destination de la Lune **(figure 2)**. D'un poids de 30 kg et d'un coût de 150 000 USD, il était composé d'un « processeur » d'environ 5000 circuits intégrés primitifs, et disposait de 74 ko de mémoire câblée (programme) et de 4 ko d'une sorte de mémoire vive. Un cas d'utilisation non prévu par les ingénieurs a causé un problème à un moment crucial de la mission Apollo 11 [3]. Visiblement, le thème de l'ingénierie est une constante dans l'histoire des systèmes embarqués.

Jusqu'au début des années 80, l'industrie utilisait des ordinateurs non connectés et équipés d'un seul processeur – tels que le PDP-11 de DEC, un représentant typique de ces ordinateurs industriels des premiers jours – mais pas encore de systèmes embarqués. Puis, au cours de cette même décennie, les automates programmables sont apparus. Très robustes et équipés d'interfaces entrées-sorties facilement utilisables, ils ont connu un déploiement rapide dans le monde industriel.

A la même époque sont également apparus les DSP (Digital Signal Processor). Très efficaces pour les opérations de base



Wikipedia [2]

Figure 2 L'interface utilisateur de l'AGC (Apollo Guidance Computer).

de traitement du signal, ces processeurs étaient au cœur d'applications hautement spécialisées, notamment dans le domaine militaire ou le domaine médical.

Dans les années 90, des processeurs relativement performants sont devenus suffisamment bon marché pour être embarqués dans beaucoup de systèmes. Enfin, les systèmes embarqués sont devenus, dès le nouveau millénaire, de plus en plus sophistiqués et se sont spécialisés pour des contextes particuliers, permettant ainsi la révolution digitale.

Aujourd'hui, les systèmes embarqués évoluent vers des systèmes standardisés et multiprocesseurs, capables de s'auto-organiser, et même de se reconfigurer pour s'adapter à leur contexte, formant ainsi une sorte d'écosystème. Ce développement est rendu possible par la forte évolution des circuits intégrés programmables. En même temps, les frontières entre systèmes embarqués traditionnels, automates programmables et systèmes à base de DSP deviennent de plus en plus floues.

L'évolution du développement du logiciel embarqué

La principale préoccupation des développeurs de logiciel embarqué de la première heure était probablement de parvenir à implémenter la fonctionnalité voulue dans un système disposant de ressources très limitées. Programmer était une activité fastidieuse pour plusieurs raisons: la programmation s'effectuait dans le meilleur des cas en assembleur et les environnements de développement étaient rudimentaires.

Avec l'apparition des systèmes d'exploitation multitâches à interfaces graphiques au début des années 90, la qualité des outils de développement a fait un grand bond en avant et l'attention des ingénieurs pouvait enfin s'orienter vers les vraies questions: celles concernant les systèmes à développer.

Dans les années 2000, par contre, la situation du développement du logiciel embarqué est devenue de nouveau moins favorable. Sous l'impulsion des énormes progrès du matériel et de l'apparition de la connectivité, la complexité du logiciel s'est fortement accrue et a posé des problèmes de maîtrise.

Cette complexité imposait une approche du développement entièrement nouvelle, à laquelle ni les développeurs expérimentés (une rupture dans les méthodes était nécessaire), ni les jeunes ingénieurs (la méthodologie n'était pas intégrée dans l'enseignement), n'étaient préparés. Comme l'illustre la **figure 3**, il en a résulté une divergence croissante entre le savoir-faire des ingénieurs et les besoins du marché.

Le rôle des systèmes d'exploitation embarqués

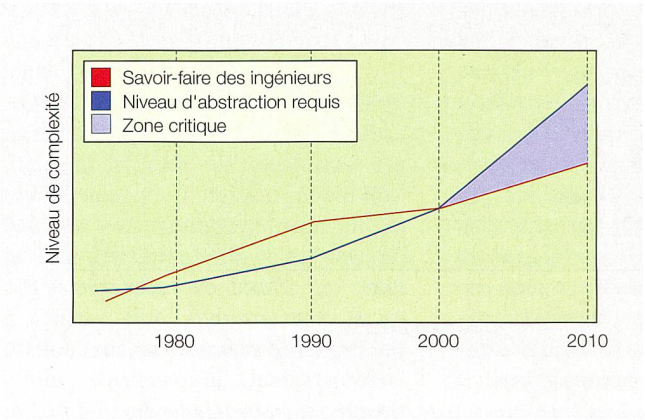
Les premiers systèmes d'exploitation embarqués sont apparus dans les années 80 déjà. Des grandes sociétés comme Wind River et Ready Systems offraient alors des systèmes d'exploitation de qualité industrielle. Cependant, leur prix élevé a limité leur déploiement à des applications industrielles conséquentes.

La révolution IT (Information Technology) du début des années 90 a, entre autres, provoqué l'émergence d'une foule d'OS (Operating Systems) pour systèmes embarqués devenus abordables pour les PME (petites et moyennes entreprises).

Toutefois, le paysage a changé dès le tournant du millénaire: un grand nombre de systèmes d'exploitation embarqués simples n'ont pas suivi l'évolution du matériel, et leur degré d'utilité a décliné, une analogie intéressante avec la situation de la **figure 3**. Les systèmes d'exploitation performants, offrant un support professionnel, ont donc repris l'avantage sur les OS libres, et une tendance actuelle consiste à utiliser des OS classiques (Linux, Windows) sur les plates-formes embarquées. Il est à noter que la plupart des systèmes embarqués n'ont pas de système d'exploitation à proprement parler et se contentent d'un simple IDF (Interrupt Driven Framework) qui met à disposition des fonctions de base, comme l'accès à la mémoire, des «timers» et des événements, fonctions permettant l'exécution pseudo-parallèle des processus du logiciel embarqué.

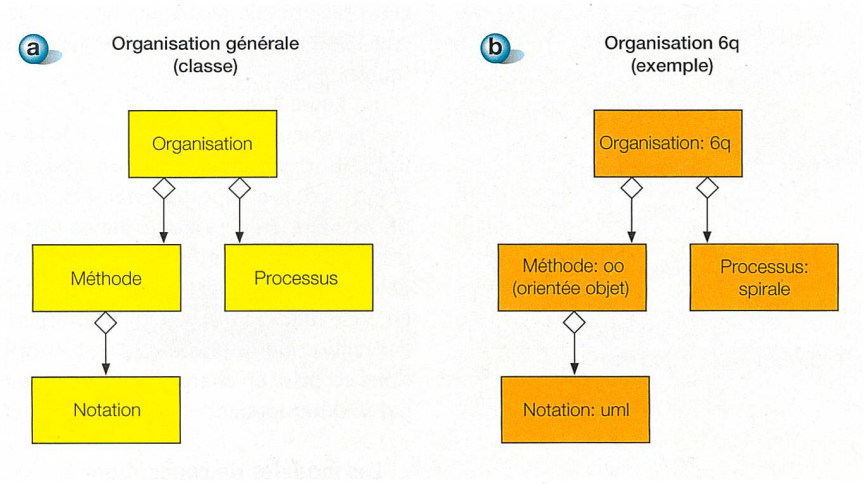
Le développement du logiciel embarqué

Quels sont donc les éléments importants dans le processus complexe du développement du logiciel embarqué? Si les réponses sont différentes pour chaque projet, on peut néanmoins esquisser des éléments communs, qui ont un impact non négligeable sur la qualité de son développement. Les paragraphes suivants les présentent de manière succincte.



HES-SO Valais

Figure 3 Situation critique due à une complexité croissant plus rapidement que les connaissances des ingénieurs.



HES-SO Valais

Figure 4 (a) Modèle général d'une organisation. (b) Exemple d'organisation compatible avec ce modèle, l'organisation « 6q ».

Une formation raisonnable

Le rôle d'une formation convenable et régulière des développeurs ne doit pas être sous-estimé. Elle est un excellent moyen pour éviter la situation présentée dans la **figure 3**. Dans les PME en particulier, organiser une telle formation n'est pas toujours facile (manque de temps et de moyens financiers). Il n'est pas rare que cet état de fait induise le gel des technologies (programmation en assembleur sur un processeur 8 bits) et contraigne à réaliser un saut radical (programmation en C++ sur un processeur 32 bits) sans que le savoir-faire nécessaire ne soit disponible (programmation orientée objet, technologie 32 bits).

Pour autant qu'elles disposent de l'avance technologique nécessaire, les écoles d'ingénieurs peuvent jouer un rôle significatif dans la formation de base, ainsi que dans le conseil aux PME et dans la formation continue des ingénieurs.

L'organisation du développement

Dans une approche professionnelle, il est nécessaire d'appliquer une organisation du développement. L'important n'est pas la forme de l'organisation en soi, mais le fait que tous les développeurs connaissent leur rôle dans cette organisation et le remplissent.

La **figure 4** présente les éléments d'une organisation générique. Une équipe de développement doit partager une même manière d'analyser et de concevoir (méthode) ainsi que de documenter (notation). L'utilisation d'une méthode et d'une notation communes évite des problèmes de compréhension et de communication au sein de l'équipe.

Autre élément important, le processus de l'organisation décrit le déroulement d'un projet. Il n'inclut pas seulement la séquence des étapes, mais aussi l'association des développeurs aux étapes. On peut ainsi éviter un piège classique : confier la responsabilité de toutes les étapes d'un projet à une même personne. Des personnes différentes doivent s'impliquer dans la spécification, l'implémentation et l'intégration. Typiquement, il serait souhaitable d'impliquer des personnes ayant une vue plutôt « client » dans les phases de spécification et d'intégration. L'approche du projet sous des angles différents contribue à la stabilité du résultat final.

La **figure 5** montre les étapes de base du processus, avec une séquence itérative et incrémentale (spirale). Elle met également en évidence le rôle important d'une méthode et d'une notation communes dans l'organisation : grâce à elles, les informations transitent sans dégradation, d'une étape à l'autre, et entre les développeurs.

Une architecture commune

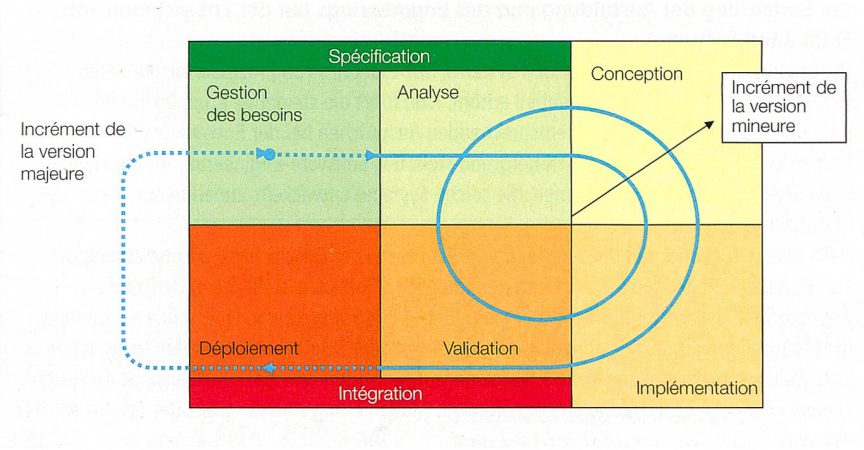
Capacité d'évolution et portabilité sur différentes cibles (processeur et système d'exploitation) sont deux objectifs importants pour les équipes de développement. Ils sont atteints grâce au découplage des différentes fonctions du programme.

L'architecture en couches permet justement d'effectuer ce découplage. Dans ce type d'architecture, chaque couche est responsable d'implémenter et d'offrir un certain nombre de fonctions accessibles à travers une interface bien décrite. L'expérience permet d'identifier un certain nombre de fonctions qui peuvent être classées en trois familles :

- Fonctions d'un OS.
- Accès à des ressources matérielles.
- Accès à des bibliothèques tierces.

Dans la partie (a) de la **figure 6** sont représentés de façon schématique trois programmes embarqués implémentés de manière monolithique dans des produits différents. Dans la partie (b), ces mêmes programmes sont représentés, mais conçus cette fois avec une architecture en couches.

La **figure 7** montre une architecture en couches simple, applicable au développement de logiciels embarqués. La couche d'application repose sur les couches XF (Execution Framework), HAL (Hardware Abstraction Layer) et LAL (Library Abstraction Layer). La couche HAL abstrait le matériel et présente une interface unique pour toutes les couches suivantes. C'est l'élément-clef des développements portables. La couche LAL est optionnelle et uniquement nécessaire si l'on utilise des produits de tiers sous forme de bibliothèques. Un produit de tiers peut alors être remplacé par un autre uniquement



HES-SO Valais

Figure 5 Les étapes de base d'un processus d'organisation.

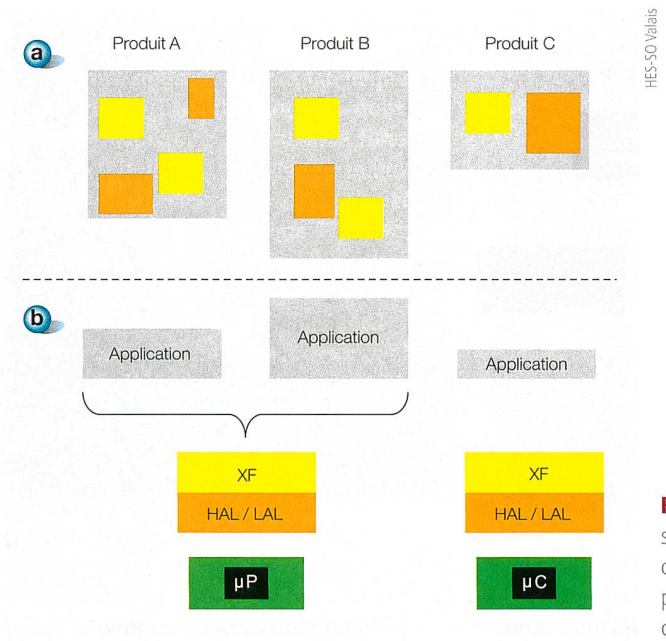


Figure 6 Produits sans architecture (a) en comparaison avec des produits à architecture commune (b).

par adaptation de la couche LAL. Finalement, la couche XF abstrait un éventuel système d'exploitation embarqué, ou alors représente elle-même un système d'exploitation minimal de style IDF. Elle offre toutes les fonctions nécessaires à l'exécution des machines d'états-transitions et, éventuellement, des utilitaires pour les structures de données, etc.

On croit souvent, à tort, que l'introduction d'une architecture est trop gourmande (code plus grand et moins rapide). Mais sans architecture, le manque de structuration engendrerait un haut niveau de complexité et le code, très monolithique, serait extrêmement difficile à faire évoluer.

L'introduction d'une architecture peut certes influer sur la taille du code et sa vitesse d'exécution, mais elle présente des avantages imbattables : standardisation d'une grande partie du logiciel des différents produits, stabilité augmentée, réutilisabilité maximale du code développé et, par conséquent, temps de développement réduit. De plus, les développeurs peuvent se concentrer sur la logique et les besoins particuliers du domaine, puisqu'ils ne sont pas distraits par des problèmes technologiques.

La modélisation

La modélisation contient deux éléments indispensables à tout développement de systèmes embarqués : l'abstraction et la spécification.

L'abstraction est l'élément primordial pour gérer la complexité du logiciel embarqué. Elle est également étroitement couplée avec la méthode utilisée et avec sa notation : la méthode détermine comment un système est abstrait et la notation permet l'expression précise de l'abstraction.

La spécification est un élément critique, car elle permet de comprendre l'ensemble d'un système. Elle fournit également la documentation nécessaire pour ceux qui vont implémenter le système, et les prescriptions de test qui vont servir à la validation. Ainsi la spécification, comme les étapes ultérieures, est intimement liée au fait qu'une organisation et un processus ont été définis et appliqués.

Enfin, la modélisation doit rester au niveau du système et ne doit pas traiter des problèmes d'implémentation. Pour cette raison, elle doit s'appuyer sur des couches technologiques (XF, etc.) préexistantes.

La **figure 8** montre, grâce à un exemple, comment la modélisation peut améliorer le développement d'un système. D'une part, le comportement est exprimé de manière très parlante **(figure 8a)**, et d'autre part, la notation univoque du modèle permet une traduction automatique en code **(figure 8b)**. Un détail important : l'exécution de la machine d'états-transitions est prise en charge par le XF et non par le développeur.

Les modèles de conception

Les modèles de conception (pattern), malheureusement trop méconnus, permettent de réduire l'effort de développement. Il s'agit d'une solution élaborée pour un problème type. Il suffit donc que les développeurs identifient un problème de conception, en trouvent le modèle de conception et l'appliquent de manière mécanique. Il en résultera un gain en temps et en stabilité.

La traduction en code du modèle d'une machine d'états-transitions est un exemple de modèle de conception. Par ailleurs, l'organisation en couches est elle-même un modèle de conception connu. Faisons donc l'expérience de l'application de modèles de conception !

Pensées finales

Les éléments suivants contribuent à la réussite du développement de systèmes embarqués :

- La veille technologique – dans le domaine du matériel comme dans celui du logiciel – prévient la formation d'un fossé trop important entre les technologies ef-

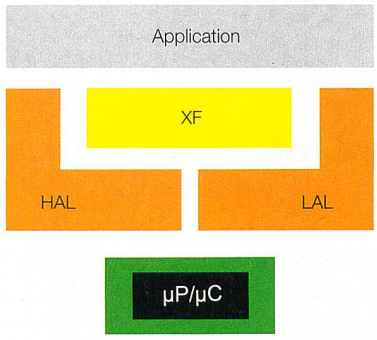
Zusammenfassung **Embedded Software**

Die Bedeutung der Ausbildung und des Engineerings bei der Entwicklung von Embedded Software

Die Leistungsfähigkeit von Embedded Systems, insbesondere von Prozessoren, die deren Herzstück bilden, hat sich exponentiell erhöht. Nun führt die steigende Komplexität der Hardware gleichzeitig jedoch zu entsprechenden Ansprüchen bei der Entwicklung der Embedded Software. Vor diesem Hintergrund stellt das Software-Engineering für die kleinen und mittelständischen Unternehmen, die solche Systeme entwickeln, zunehmend eine Herausforderung dar.

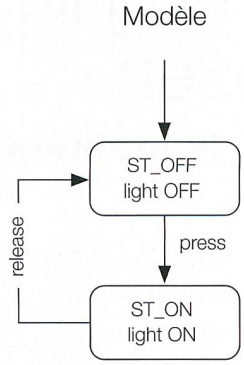
Nach einem Rückblick auf die verschiedenen Schritte in der Geschichte der Entwicklung der Computer, der Embedded Systems und der Embedded Software, stellt dieser Artikel die wesentlichen Elemente zur Entwicklung von Embedded Software vor. Eine solide Ausbildung (Grundausbildung und Weiterbildung) der Ingenieure, die Schritt halten mit den technischen Entwicklungen, die Nutzung einer Entwicklungsorganisation und eines Aufbaus mit mehreren Ebenen sowie die Anwendung einer geeigneten Modellierung sind die Basis des Erfolgs solcher Projekte.

CHe



HES-SO Valais

Figure 7 Une architecture simple et efficace.



Code

```
enum t_state {st_on, st_off};
t_state state;
state = st_on;
switch state
{
    case st_on: if (ev_press)
                state = st_off;
    case st_off: if (ev_release)
                state = st_on;
    default: state = st_off;
}
if (state == st_on) light_on()
else light_off();
```

HES-SO Valais

Figure 8 Comparaison de lisibilité d'un modèle et du code.

fectivement appliquées et celles qui sont disponibles. Elle permet d'anticiper le renouvellement de celles qui arrivent aux limites de leurs possibilités.

- La définition et l'application d'une organisation de développement, même très simple, aide à mieux coordonner les différents acteurs et leurs activités.
- La mise en place d'une architecture de logiciel fait apparaître des éléments communs, ce qui réduit grandement l'effort de création de nouveaux produits, et augmente la stabilité des produits et la compétitivité de l'entreprise.
- La modélisation assure la conformité des produits, la qualité de la documentation technique et l'établissement de prescriptions de validation.
- L'application de modèles de conception garantit une implémentation efficace.

Il est important que tous les acteurs concernés – ingénieurs, industries et écoles – soient conscients de ces éléments, afin qu'ils puissent assumer leur rôle dans le maintien du niveau professionnel nécessaire au développement embarqué de demain.

Le futur

Les grands acteurs du monde de l'« embarqué » ne vont pas cesser d'innover. Les PME ne pourront pas se permettre de laisser partir le train, un sacré défi !

Du point de vue matériel, les systèmes multicœurs, les processeurs à consommation ultra-basse et les systèmes reconfigurables vont prendre la main. Du point de vue logiciel, le développement basé modèle devra prendre en compte ces développements matériels.

Les moyens sophistiqués de communication vont permettre l'émergence de deux autres tendances : l'auto-organisation des systèmes embarqués entre eux, et leur intégration verticale dans les systèmes IT classiques. Dans ce contexte, l'importance de la formation et de l'ingénierie va encore s'accroître.

Une spirale sans fin ? Peut-être, oui, mais même sous cet angle, le développement de systèmes embarqués reste une belle aventure avec des challenges exaltants pour l'industrie et pour la formation.

Références

[1] http://en.wikipedia.org/wiki/Moore's_law.
[2] http://en.wikipedia.org/wiki/Apollo_Guidance_Computer.
[3] http://en.wikipedia.org/wiki/Apollo_Guidance_Computer#PGNCS_trouble.

Informations sur les auteurs

Medard Rieder est ingénieur ETS en mécanique. Il est aujourd'hui professeur à la HES-SO Valais. Il donne des cours de formation de base et de formation continue dans le domaine du génie logiciel pour systèmes embarqués. Ses projets avec l'industrie lui permettent d'une part d'optimiser ses cours et, d'autre part, de constater les préoccupations de PME confrontées à une évolution technologique au rythme débridé.

HES-SO Valais, 1950 Sion, medard.rieder@hevs.ch

Dominique Gabioud est ingénieur EPF en électricité. Dans son activité de professeur de télécommunications à la HES-SO Valais, il s'intéresse en particulier aux systèmes de contrôle et de gestion liés au monde de l'énergie.

HES-SO Valais, 1950 Sion, dominique.gabioud@hevs.ch

Rico Steiner est ingénieur HES en systèmes industriels. Il est aujourd'hui adjoint scientifique auprès de la HES-SO Valais. Il contribue notamment à des projets de recherche appliquée dans le domaine de l'ingénierie logicielle pour systèmes embarqués.

HES-SO Valais, 1950 Sion, rico.steiner@hevs.ch

Anzeige

Fehlerfreie Texte ...?

KOMMAZWERG

Korrekturbüro Kommazwerg, www.kommazwerg.ch, kontakt@kommazwerg.ch
Petra Winterhalter, eidg. dipl. Korrektorin, Tel. +41 76 592 31 29