(S43b - CLIPS)

(Jonas Loy)
(Jean-Baptiste Murphy)

(Octobre 2016)

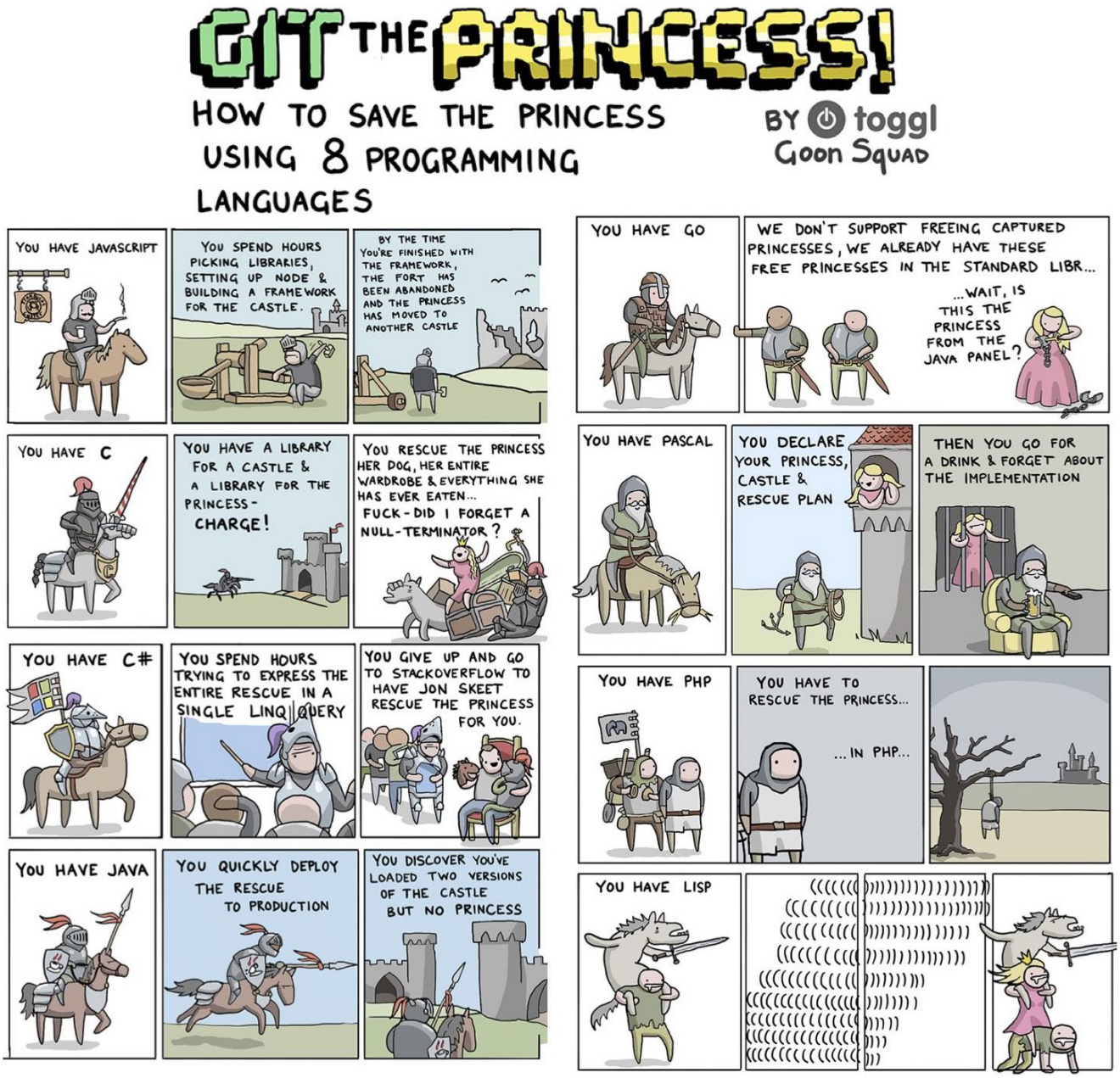

Figure 1: (=(+ 1 1)2)

1 (Structure d'implémentation)

1.1 (Les différentes géométries)

Plusieurs géométries ont été utilisées pour pour pouvoir caractériser notre pièce et ont été définies dans les "deftemplate" du code. Les trous et plans ont été définis comme dans le sujet. Les slots/pockets, quant à eux, ont été définis de manière légèrement différente afin de simplifier la définition de la pièce et des règles.
Les éléments rajoutés au template sont:

- **(Inverse or NonInverse)**: permet de savoir si un slot est défini comme inversé, c'est-à-dire s'il faut considérer le "négatif" du slot défini.
- **(BottomOrientation & SideOrientation)**: permettent de définir la direction normal au fond du slot (Bottom) et celle normale à ces côtés les plus longs. (Cet ajout se révèle utile pour la définition des règles de direction de machine pour les slots/pockets.

Sont également définis les relationships entre les features (à l'aide du deftemplate Relationships).

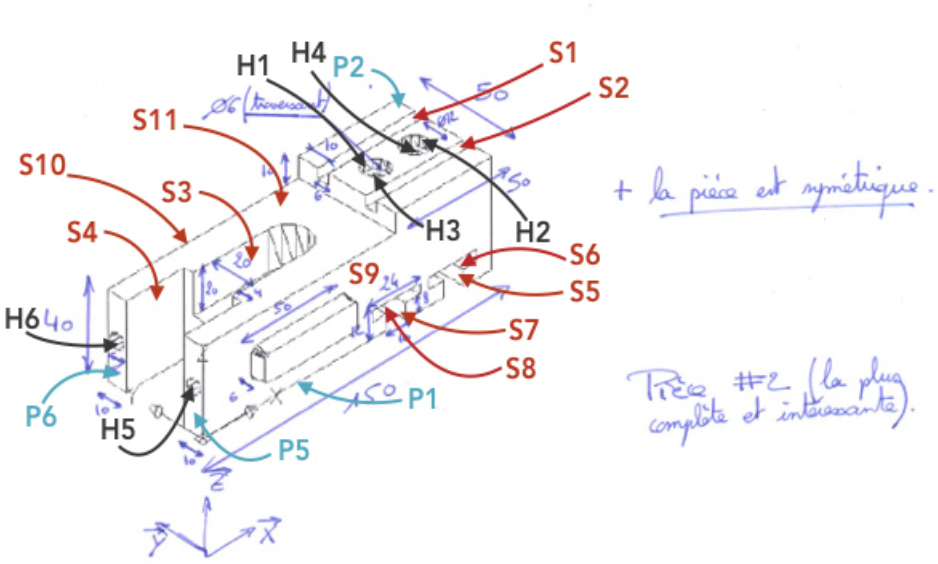

Figure 2: (Détail des géométries)

1.2 (Les machines-outils)

Les derniers templates sont associés aux machine et outils d'usinage:

(Tool) désigne un outil par son diamètre et sa longueur

(Machine) représente une machine à l'aide de " orientation d'usinage.

ainsi qu'à leur lien avec les features:

(MachiningDirection) référence la direction d'usinage et le paramètre Face/Side milling d'un feature.

(FeatureMachinedBy) lie le feature à une machine

(FeatureMachinedWith) lie le feature à son outil

(phaseList) donne une gamme d'usinage possible pour chaque machine

1.3 (La définition des règles)

Ayant modifié le template des features, il a fallut apporté une mise à jour aux règles proposées dans le sujet.

A l'étape 3, deux règles ont été ajoutées (Inverse Side/Face Milling Pocket) concernant les pocket avec l'attribut face ou side milling qui permettent dans le cas où le pocket est "Inverse" de ne pas se soucier du diamètre de l'outil.

On peut noter que grâce à notre implémentation, certaines règles comme celle sur la perpendicularité des plans ou celle sur les Round Through Pockets ne sont pas utilisées ici, simplifiant ainsi le code.

Enfin une salience a été rajoutée pour chaque règles mais n'a pas d'impact ici étant donné qu'elle respecte l'ordre chronologique de définition des règles. Nous l'avons tout de même laissé au cas où on souhaiterait rapidement changer le poids d'une règle.

2 (Remarques sur la partie 4)

Dans cette étape il s'agit de donner les phases d'usinage des pièces.

Pour cela nous avons établi une liste non-ordonnée (elle le sera dans la partie suivante) de type *multifield* qui donne une possibilité de sous-phase d'usinage.

Cependant, cette sous-phase n'est pas encore définitive : puisque que jusqu'alors nous voulions un résultat le plus général possible, nous avons plusieurs direction d'usinage pour chaque feature. Chaque feature pourra donc apparaître dans plusieurs sous phases ou apparaître plusieurs fois dans la même sous-phase.

Cela n'est pas vraiment un problème et est même souhaité à ce stade car nous obtenons une réponse générale qui peut mener à un choix pertinent et adapté à la situation de l'entreprise. Le regroupement en sous-phases définitif peut être le fruit du travail d'un ingénieur, ou, d'une règle conçue à cet effet.

Cependant, ce travail de regroupement n'est pas aussi automatique que les tâches que nous avons implémenté précédemment. Il dépend énormément des conditions de production de la pièce : une usine produisant des pièces pou l'automobile n'aura pas la même organisation qu'une usine produisant des prototypes. Pour obtenir des résultats satisfaisant, on pourra s'appuyer sur les critères suivants:

- La minimisation de l'investissement nécessaire (qui inclut le nombre de machines, de postes de lavage, de supports d'usinages, de convoyeurs, de machines de contrôle etc)
- L'optimisation du temps de chaque sous-phase pour se rapprocher le plus possible du TAKT Time.
- La minimisation de l'espace occupé par la ligne de production.
- etc.

(Nous avons réalisé cette démarche (sans l'aide de Clips (désolé pour les parenthèses imbriquées(!))) pour le projet COSYP de M. Fromentin à Cluny en 2e année)

3 (Remarques sur la partie 5)

Il s'agit ici d'ordonner les listes précédentes.

Si les relations d'antériorités d'usinage ne sont pas respectées, nous les forçons en plaçant l'entité à réaliser en priorité au début de la liste.

Remarque : Nous avons fait le choix de ne pas supprimer les doublons dans chaque sous-phase d'usinage. Nous n'avons pas réussi à l'implémenter en utilisant les fonctions standard de manipulation de multifield (il faudrait faire deux boucles imbriquées comparant deux à deux les valeurs de la liste) et nous n'avons pas jugé cela indispensable pour ce projet.

4 (Why Lisp shouldn't exist and is a nuisance to mankind)

- Le fonctionnement des règles dépend beaucoup de la manière dont la pièce est décrite au départ. On peut imaginer qu'il est assez difficile d'adapter des règles développées pour un projet à un autre.
- Une fois qu'une règle est écrite, elle sort un résultat quoi qu'il arrive. Il est impossible de rajouter des étapes "de bon sens" au milieu des règles une fois que le programme est écrit et lancé. Par exemple, la règle FaceMilling de la Step 3 n'a pas de condition sur l'outil. Elle va donc ressortir le premier outil disponible en magasin. Il aurait sûrement été plus judicieux de comparer avec les outils les plus utilisés et d'opérer le choix en conséquence.
- Certaines règles s'avèrent compliquées à implémenter. On peut prendre l'exemple des choix des phases d'usinages car elles dépendent énormément du contexte de l'entreprise, de l'usine etc.
- On imagine que les règles vont devenir vraiment plus compliquées à implémenter si on cherche à déterminer l'outil idéal à commander à Sandvick pour réaliser une opération. Il faudrait implémenter toutes les règles du catalogue sandvick en incluant une notion de coût (en outils coupants, en temps machine, etc)

5 (Pourquoi c'est bien)

- Une fois que les raisonnements des ingénieurs sont implémentés dans des règles, on obtient rapidement des solutions d'usinage correspondant aux règles données.
- L'utilisation de clips donne un framework assez complet pour réaliser ce genre d'applications.