Instructions pour le TP1 – Simulateur SIMSE

L'objectif de ces simulations est de découvrir l'impact de vos décisions concernant l'approche différente de gestion de projet nécessaire selon le cycle de vie logiciel que votre employeur utilise. Conséquemment vous allez comprendre que pour le gestionnaire de projet logiciel, le choix d'un cycle de vie a un grand impact sur sa manière de gérer un projet logiciel. Donc, prenez le temps de lire et de vous documenter, à l'aide des références suivantes, sur le fonctionnement particulier de chaque cycle de vie. Ainsi en vous documentant sur leur fonctionnement vous allez mieux comprendre comment avoir du succès dans le jeu de simulation pour chaque cas de figure.

1.1 Références pour le cycle de vie en Cascade :

1. Boehm, B.W., *Software Engineering Economics*. 1981, Upper Saddle River, NJ: Prentice Hall, Inc.
2. Brooks, F.P., *The Mythical Man-Month: Essays on Software Engineering*. 2 ed. 1995, Boston, MA: Addison-Wesley. 336.
3. Bryan, G.E., *Not All Programmers are Created Equal*, in *Software Engineering Project Management*, R.H. Thayer, Editor. 1997, IEEE Computer Society: Los Alamitos, CA. p. 346-355.
4. Curtis, B., H. Krasner, and N. Iscoe, *A Field Study of the Software Design Process for Large Systems.* Communications of the ACM, 1998. **31**(11): p. 1268-1287.
5. Sackman, H., W.J. Erikson, and E.E. Grant, *Exploratory Experimental Studies Comparing Online and Offline Programming Performance.* Communications of the ACM, 1968. **11**(1): p. 3-11.
6. Tvedt, J.D., *An Extensible Model for Evaluating the Impact of Process Improvements on Software Development Cycle Time*. 1996, Ph.D. Dissertation, Arizona State University.

1.2 Références pour le cycle de vie incrémental :

1. Boehm, B.W., *A Spiral Model of Software Development and Enhancement.* IEEE Computer, 1988. **21**(5): p. 61-72.
2. Gilb, T., *Evolutionary Delivery versus the Waterfall Model.* ACM SIGSOFT Software Engineering Notes, 1985: p. 49-61.
3. Laman, C. and V. Basili, *Iterative and Incremental Development: A Brief History.* IEEE Computer, 2003. **36**(6): p. 47-56.
4. Royce, W., *TRW's Ada Process Model for Incremental Development of Large Software Systems*, in *Proceedings of the 12th International Conference on Software Engineering*. 1990. p. 2-11.
5. Scacchi, W., *Process Models in Software Engineering*, in *Encyclopedia of Software Engineering*, J. Marciniak, Editor. 2001, Wiley.

1.3 Références pour le cycle de vie de prototypage rapide :

1. Gordon, V.S., Bieman, J.M. *A Rapid prototyping : Lessons Learned.* IEEE Software, 1995. **12**(1): p. 85-95.
2. Kordon, F., Luqi. *An introduction to rapid system prototyping.* IEEE Transactions on Software Engineering, 2002. **28**(9): p. 817-821.
3. Gomaa, H. and Yusuf, Y.Y., *The impact of Rapid prototyping on user requirements.* ACM SIGSOFT, 1983.
4. Hartson, H.R. Smith, E.C. *Rapid prototyping in human computer interface development*, Interacting with Computers, 1991. **3**(1): p. 51-91.

1.4 Références pour le cycle de vie RUP :

1. Kruchten, P. The Rational Unified Process: An Introduction (2nd Edition). 2000: Addison-Wesley.
2. Kruchten, P. Going over the waterfall with the RUP. Rational Edge Magazine, 2001, 16 p.
3. Ambler, S.W. *A manager's Introduction to Rational Unified process (RUP).* AmbySoft, 2005, 18 p.
4. Smith, J. *A comparison of RUP and XP.* Rational Software White Paper, 2002, 22 p.
5. Parnas, D., Clements, P.C. A rational Design process, IEEE Transactions, SE-12(2), February 1986.

1.5 Références pour le cycle de vie Extreme Programming (XP) :

1. Cockburn, A. and L. Williams, The Costs and Benefits of Pair Programming, in Extreme Programming Examined. 2001, Addison Wesley.
2. Wake, W.C., Extreme Programming Explored. 2002, Boston, MA: Addison-Wesley.
3. Juric, R. Extreme Programming and its development practices, Proceedings of the 22nd International Conference on Information Technology Interfaces, 13-16 Juin 2000, p. 97-104.
4. Ji, F. and Sedano, T. Comparing extreme programming and waterfall project results, Proceedings of the 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 22-24 Mai 2011, p. 482-486.
5. Beck, K. Embracing change with extreme programming, Computer, 32(10), Oct 1999: p. 70-77.

2- Installez SIMSE sur votre poste de travail :

2.1 MacOS :

1. Vérifiez que vous avez le Java SE Development kit 14: Java jdk-14_osx-x64.bin d'installé pour votre macOS. Sur mac la dernière version de Java fonctionne bien avec le jeu ;
2. Obtenez les fichiers (.jar) de chacun des 4 cycles de vie dans la section download du site SIMSE : Games = (waterfall, incremental, rapid prototyping, Rational Unified Process et Extreme Programming) et mettez les sur votre bureau ;
3. Démarrez un terminal :

    1. changez de répertoire avec la commande cd desktop ;
    2. démarrez un jeu avec la commande java - jar waterfall.jar ;
    3. changez **waterfall** par le nom des autres cycles de vie pour les démarrer.

2.2 Windows 10 :

1. Les dernières versions de Java ne fonctionnes pas bien avec ce jeu. Vérifiez que vous avez le Java SE 11 (LTS) Java jdk-11.0.6_windows-x64_bin.exe (le windows x64 Installer) d'installé sur votre équipement. Si vous avez une version plus récente de Java, enlevez-la et remplacez-la par la version 11.0.6 (vous pourrez la remettre après le cours);
2. Obtenez les fichiers (.jar) de chacun des 4 cycles de vie dans la section download du site SIMSE : **Games** = (waterfall, incremental, rapid prototyping, Rational Unified Process et Extreme Programming) sur votre bureau ;
3. Double-cliquez sur le fichier **.jar** et le jeu débutes.

3- Faites votre formation au jeu SIMSE en regardant les vidéos que nos étudiants ont rendu disponible sur Youtube (merci à Youssef) :

1- Débutez avec le tutoriel du simulateur **SIMSE** :
https://www.youtube.com/watch?v=lJpHPJrj6Pc&t=13s

2- Formez-vous sur l'outil de branchement SIMSE pour répéter une partie de la simulation à partir d'un point précis et prendre une autre décision :
https://www.youtube.com/watch?v=t-g2huTfzsQ&t=5s

3- Étudiez l'outil explicatif SIMSE qui permet de révéler des détails de la simulation :
https://www.youtube.com/watch?v=iiDJmBs-6UU&t=3s

4- Conservez des notes pour l'examen en ligne :

Pour ce Tp1, vous devez effectuer la simulation de tous les cycles de vie et prendre des notes personnelles qui vous seront utiles pour répondre aux questions de l'examen:

- Notez votre meilleure note obtenue et le nombre de 'ticks' (c.-à-d. jours) et notez le nombre maximum de 'ticks' à ne pas dépasser pour obtenir le meilleur score;
- Pour tous les cycles de vie, notez la séquence des étapes (stratégie) à suivre pour obtenir le meilleur résultat;
- Pour tous les cycles de vie, retenez le nom du système à développer;
- Notez l'impact de l'assignation des personnes aux tâches selon leurs compétences/expérience et d'impliquer les bonnes personnes au bon moment;
- Notez l'importance des revues pour découvrir les défauts qui se cachent dans le logiciel (tentez de voir, avec l'outil explicatif combien de défauts se cachent dans votre projet);
- Notez l'impact d'acheter ou pas des outils d'ingénierie
- Analysez les objets (par exemple: l'énergie, le 'Mood' et le 'Payrate' des employés, les caractéristiques des livrables (Nb d'erreurs détectées, Nb d'erreurs non détectées, %d'erreur, %Complété) et l'impact de l'utilisation d'outils de génie logiciel (Coût, facteur d'augmentation de la productivité et Facteur de réduction d'erreurs);
- Dans le cycle de vie waterfall :
  a) quel est l'effet de congédier André dès le début du projet?
  b) qu'est qu'un débutant comme Roger peut et ne peut pas faire comme tâche?
  c) qu'arrive-t-il si vous ne complétez pas complètement une étape et démarrez quand même l'étape suivante?
  d) qu'arrive-t-il (2 impacts) quand on met trop d'employés au travail sur une même tâche?
  e) à quel moment il est préférable d'effectuer la revue d'un artéfact?
  f) quel est l'impact de ne pas préparer les tests avant de les effectuer?
  g) comment doit-on traiter une demande de changement efficacement dans ce cycle de vie?

- Dans le cycle de vie incrémental :
    a) quel est l'effet de sauter une ou plusieurs phases de la documentation (exigences/conception) sur un ou plusieurs modules?
    b) quels sont les attributs les plus importants pour prendre des décisions?
    c) quel attribut s'améliore si on effectue les exigences avant la conception? Essayez de sauter l'étape des exigences pour voir l'effet.
    d) quel est l'impact d'une soumission précoce?
- Dans le cycle de vie prototypage rapide :
    a) essayez tous les langages et identifiez le bon langage pour prototyper versus l'implémentation finale. Et aussi regardez l'effet de l'utilisation du même langage pour ces deux activités;
    b) quel est l'effet de faire trop peu ou beaucoup trop de prototypage sur le projet?
- Dans le cycle RUP :
    a) combien de fois pouvez-vous acheter des outils?
    b) quels sont les artéfacts produits lors de l'«Inception»?
    c) quelles sont les activités principales de la phase de développement?
    d) quel est l'effet de faire trop ou pas assez d'itérations dans une phase?
    e) quelle phase prends le plus de 'ticks'?
- Dans le cycle de vie Extreme Programming :
    a) quelles sont les activités d'une réunion de planification?
    b) que se passe-t-il si vous n'incluez pas le client dans les réunions?
    c) que se passe-t-il si vous n'incluez pas un développeur dans une réunion de planification?
    d) que se passe-t-il si vous ne faites pas les tests unitaires au bon moment?
    e) à quel moment il faut faire les tests pour avoir le meilleur score?
    f) est-ce que le budget est important dans ce jeu?