Rechercher de nouvelles méthodes de détection des erreurs

Soutenu par :

El ALAOUI Imane

EL HARRAK Raghib

LAHLOU Ouidad

WICKERT Nicolas

Encadré par :

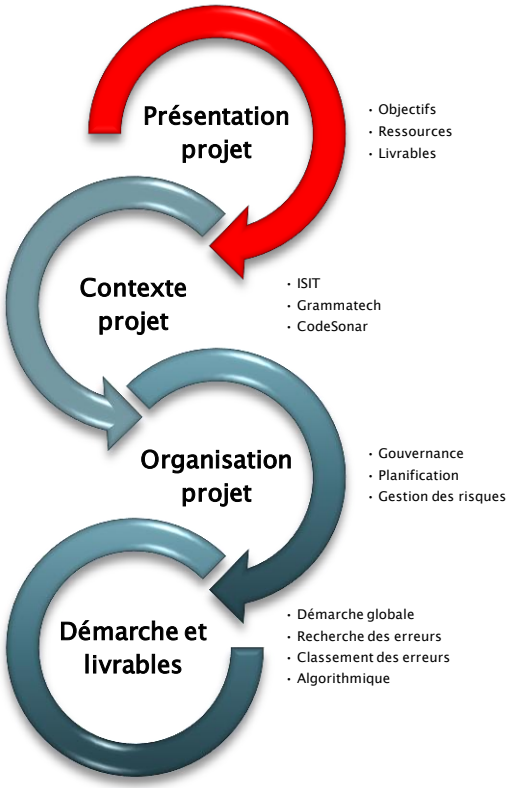
Micaël Martins

Alexis Todoskoff

2012/2013 – QSF-S

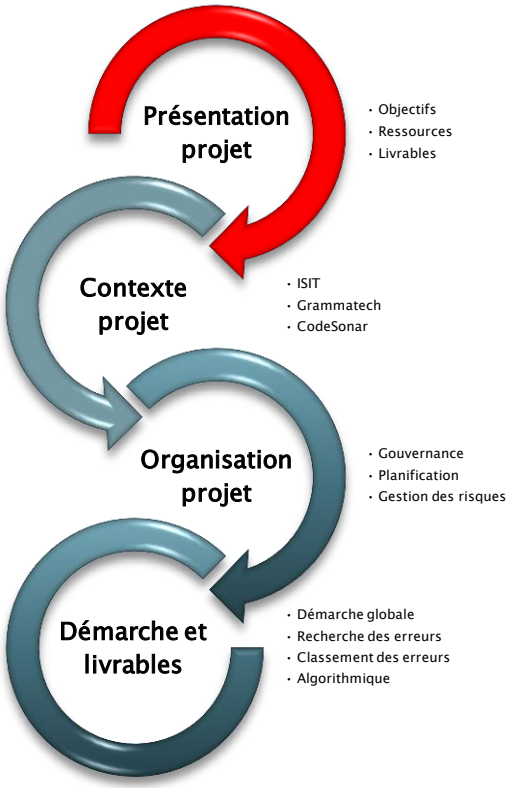
Sommaire

Objectifs



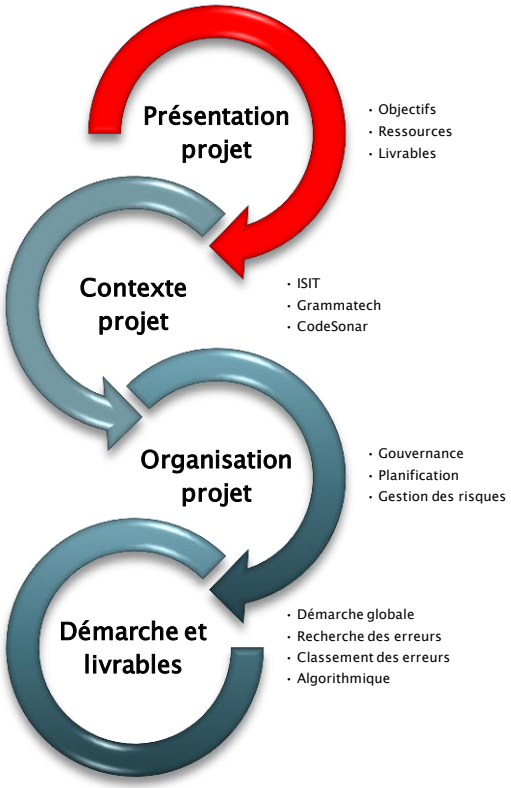
- Objectifs principaux
  - Rechercher d'erreurs intégrables à CodeSonar
  - Classifier et analyser les erreurs
  - Rechercher des solutions implémentables
- Enjeu
  - Participer à l'amélioration de l'outil CodeSonar
- Périmètres
  - Langages C et C++
  - Erreurs Safety/Security

Ressources

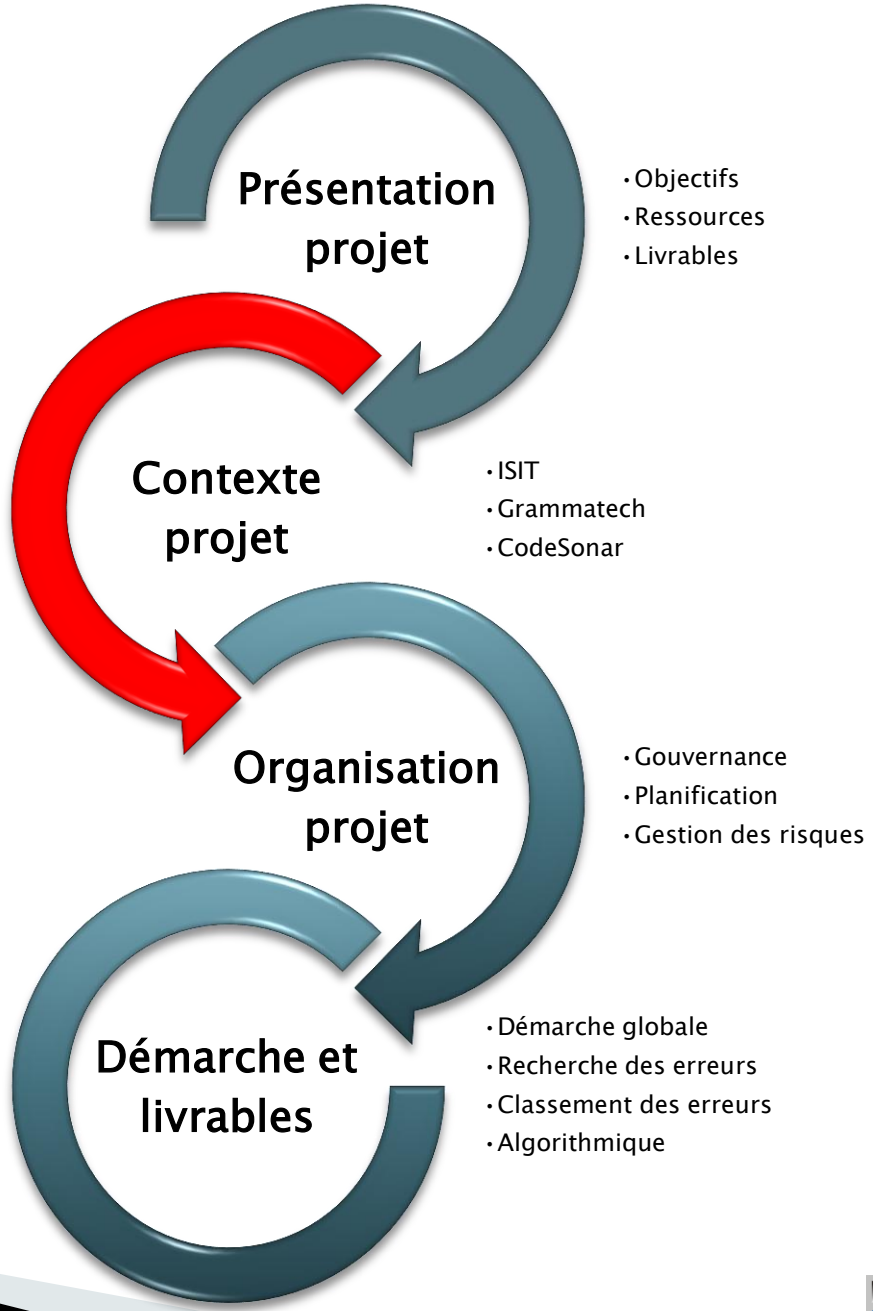


- Ressources
  - 2 étudiants ingénieur QSF-S
  - 2 étudiants en Master QUASSI
- 20 journées de projet
- Encadrement industriel et technique

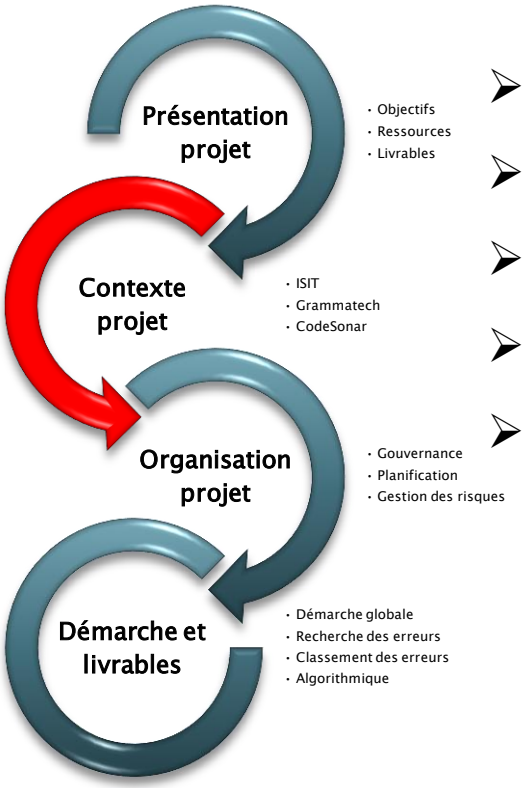
Livrables



- Livrables attendus
  - Liste classifiée des erreurs non détectées par CodeSonar
  - Réflexion sur l'implémentation d'une détection automatisée des erreurs
  - Plugin permettant de corriger certaines erreurs


Présentation projet
Objectifs
Ressources
Livrables
Contexte projet
ISIT
Grammatech
CodeSonar
Organisation projet
Gouvernance
Planification
Gestion des risques
Démarche et livrables
Démarche globale
Recherche des erreurs
Classement des erreurs
Algorithmique

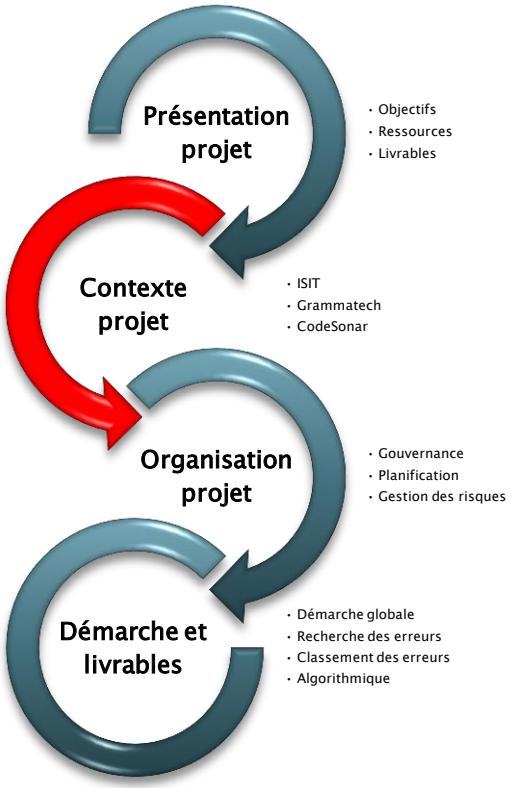
ISIT



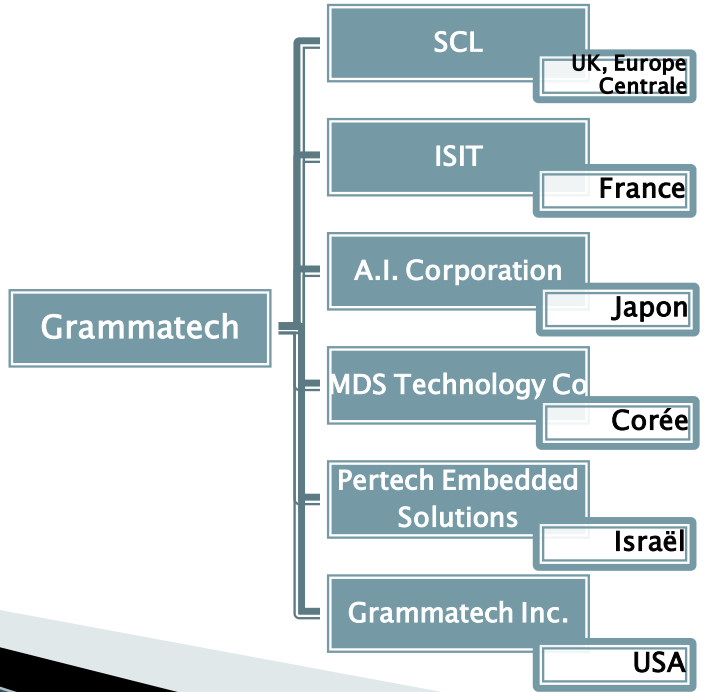
- Crée en 1991 par 2 ingénieurs
- 20 employés
- Chiffre d'affaires: 3M euros
- Capital social: 80 K euros
- Activités:
  - Systèmes embarqués
  - Génie logiciel, test & validation
  - Distributeur de Code Sonar
  - Automatisme
  - Test de carte & programmation

Test de carte électronique

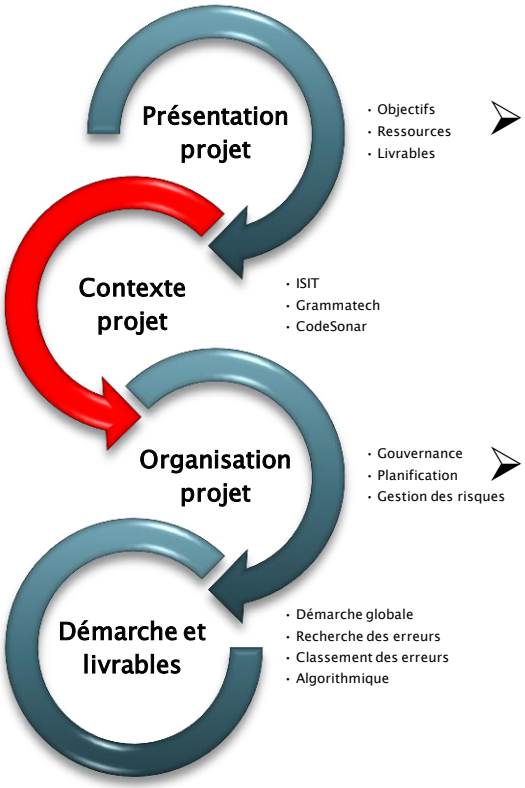
Grammatech



- Société américaine, basée à Ithaca
- Fondée en 1988
- Prix de l'impact rétrospectif 2011 (SIGSOFT)
- Editeur de CodeSonar



CodeSonar

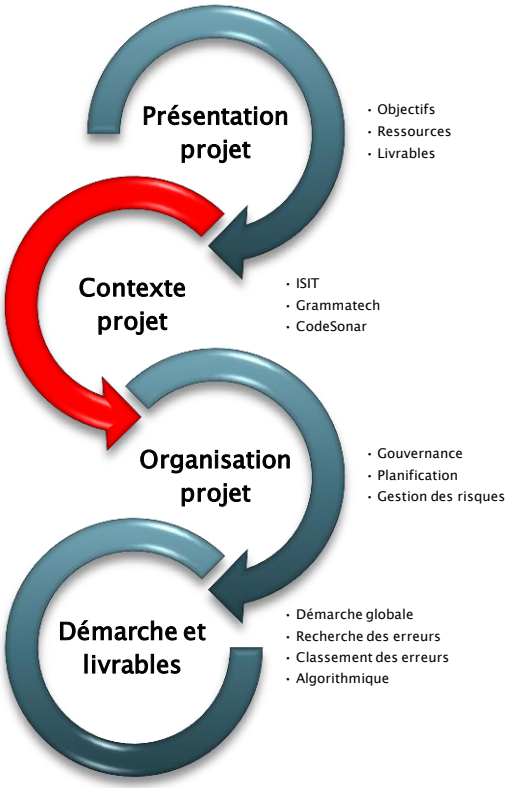


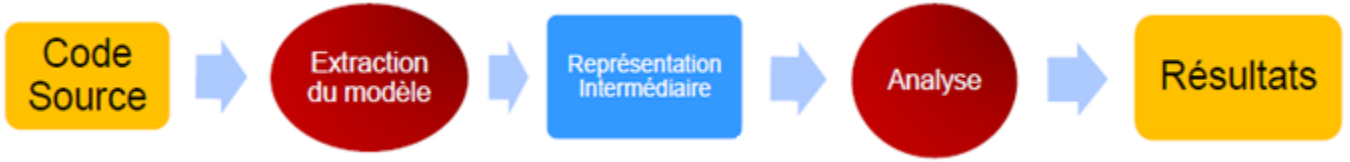
- Logiciel d'analyse statique et arbitraire de code C/C++
  - Inspiré du logiciel Sonar
  - Interface web

CODESONAR

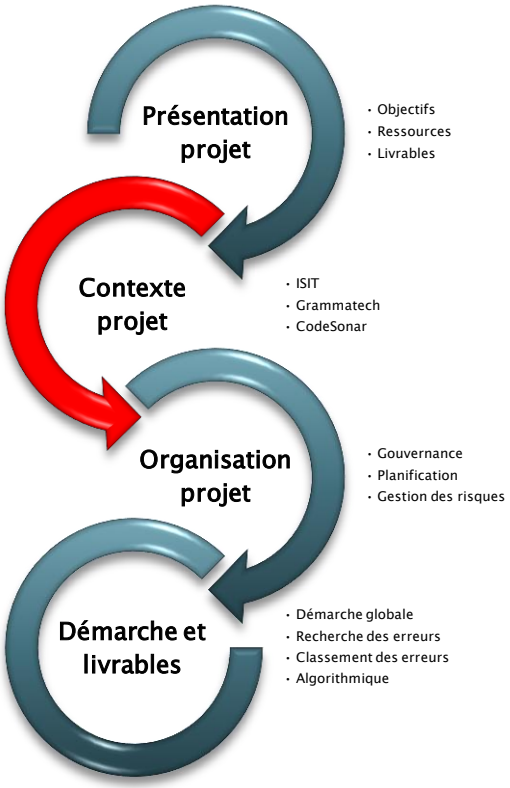
- Les erreurs détectables par CodeSonar se répartissent en 4 catégories:
  - Vérifications basiques
  - Vérifications sur les librairies C standards
  - Vérifications optionnelles
  - Vérifications définies par l'utilisateur

CodeSonar


Présentation projet
Objectifs
Ressources
Livrables
Contexte projet
ISIT
Grammatech
CodeSonar
Organisation projet
Gouvernance
Planification
Gestion des risques
Démarche et livrables
Démarche globale
Recherche des erreurs
Classement des erreurs
Algorithmique


Code Source
Extraction du modèle
Représentation Intermédiaire
Analyse
Résultats

CodeSonar



findutils-4.2.27 analysis 1 : findutils-4.2.27 : CodeSonar - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:9000/analysis/4590.html

Home | Tips | Manual | About | Sign in

CODESONAR Search this analysis for Search Warnings | Advanced Search

Home › findutils-4.2.27 › findutils-4.2.27 analysis 1 CSV | XML | Visible Warnings: active

findutils-4.2.27 : findutils-4.2.27 analysis 1 < previous next >

Analysis: findutils-4.2.27 analysis 1 edit
Analysis Description: *None.* edit
Analysis ID: 4590
Started: 01/11/11 17:20:18 show details
Finished: 01/11/11 17:49:51 show logs
Analysis State: Finished
Files: 128 total view all
Lines With Code: 26,998
Warnings: 324
Charts and Tables: show options

Remove this analysis.

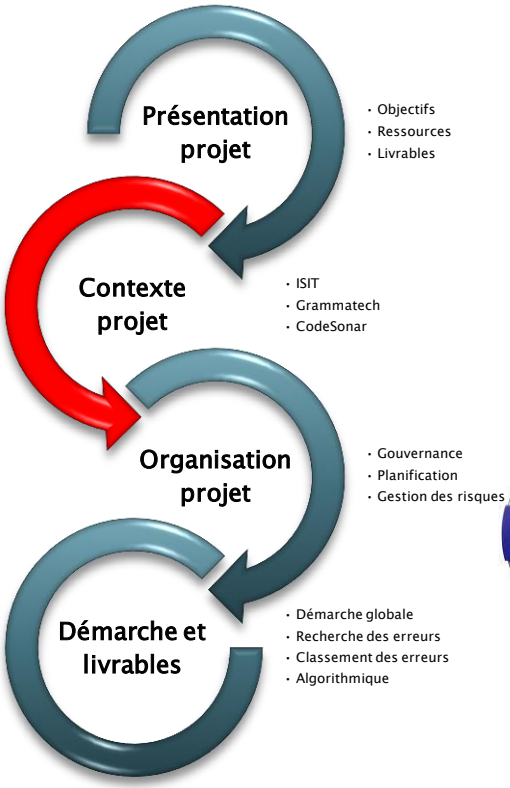
Warnings

|<< < 1 - 50 of 324 > >>|
Goto Show More Show Fewer

ID	Class	Rank	File	Line Number	Procedure	Priority	State	Finding Owner
68811.582477	Leak	20.56	pred.c	1525	prep_child_for_exec	None	None	None
68708.582367	Leak	21.29	regcomp.c	3513	build_charclass_op	None	None	None
68707.582366	Leak	21.29	regcomp.c	3511	build_charclass_op	None	None	None
68714.582373	Leak	21.29	regcomp.c	2986	parse_bracket_exp	None	None	None
68713.582372	Leak	21.29	regcomp.c	2984	parse_bracket_exp	None	None	None
68768.582434	Leak	21.29	regexec.c	517	re_copy_regs	None	None	None

Done

CodeSonar



- Objectifs
- Ressources
- Livrables

- ISIT
- Grammatech
- CodeSonar

- Gouvernance
- Planification
- Gestion des risques

- Démarche globale
- Recherche des erreurs
- Classement des erreurs
- Algorithmique

SAMSUNG

ArcelorMittal

NASA

AIRBUS

UNITED STATES NAVY

VOLVO

UCL

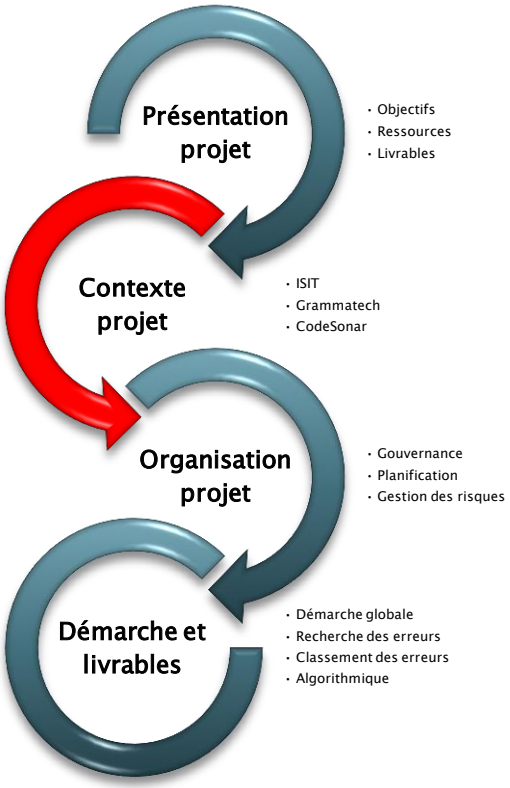
FDA

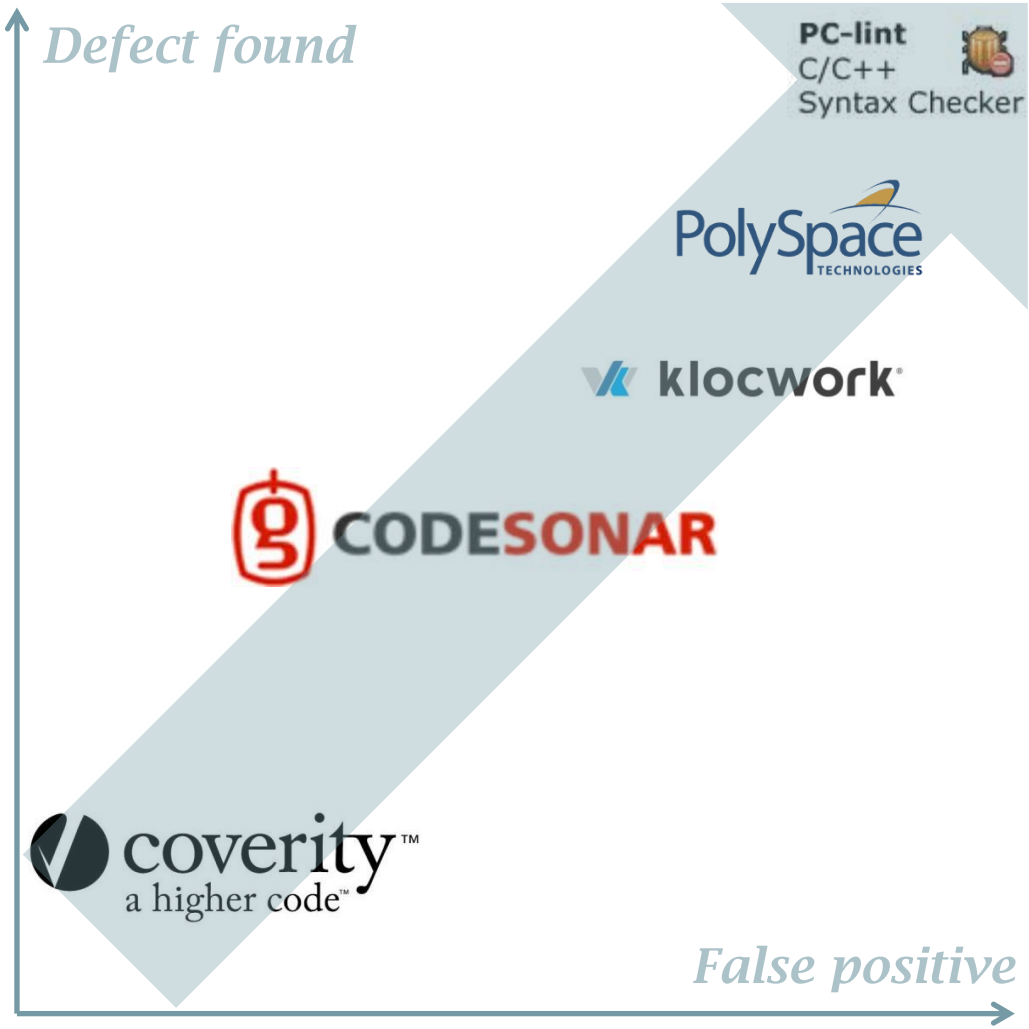
PHILIPS

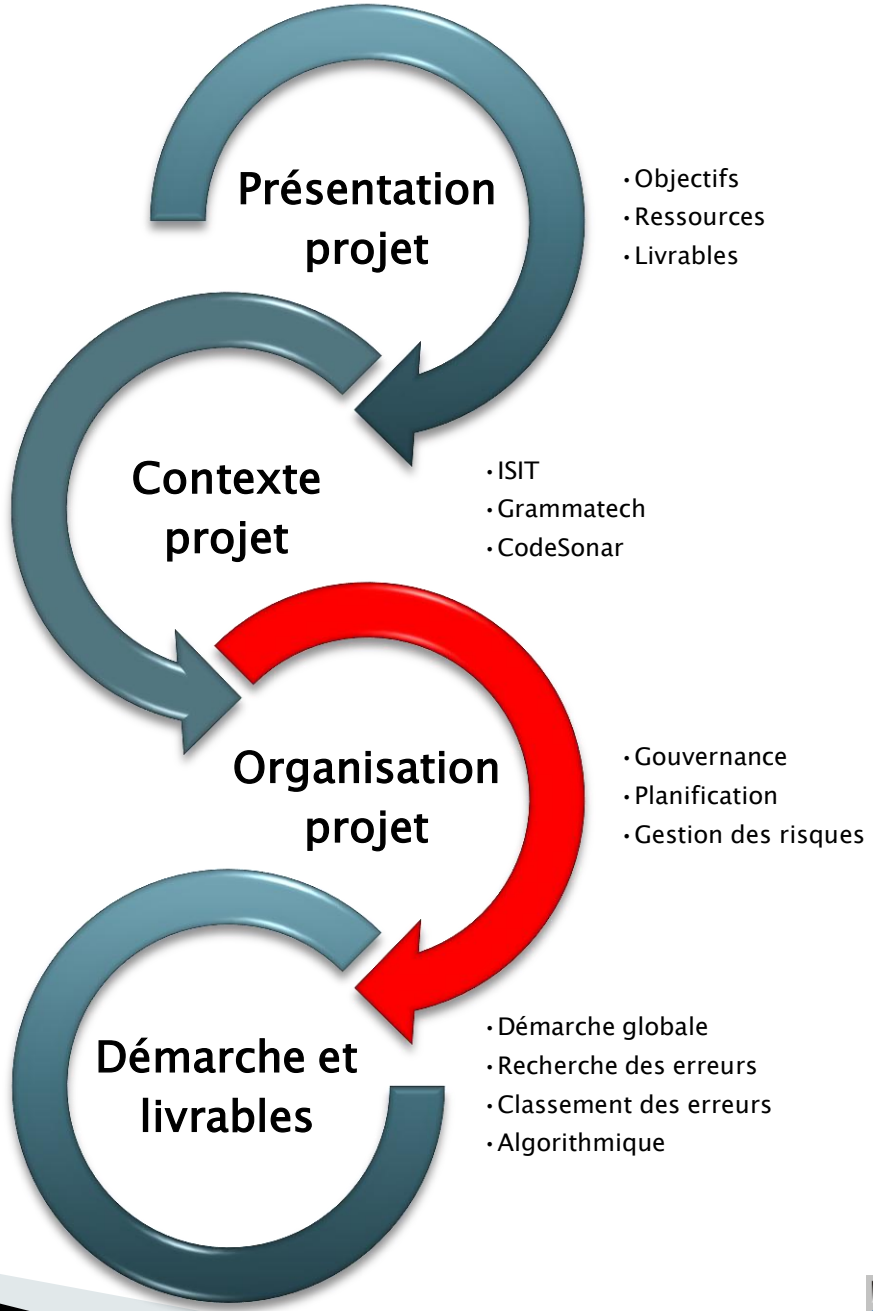
SOLTAMEDICAL
THERMAGE | ISOLAZ | FRAXEL
CLARO

U.S.ARMY

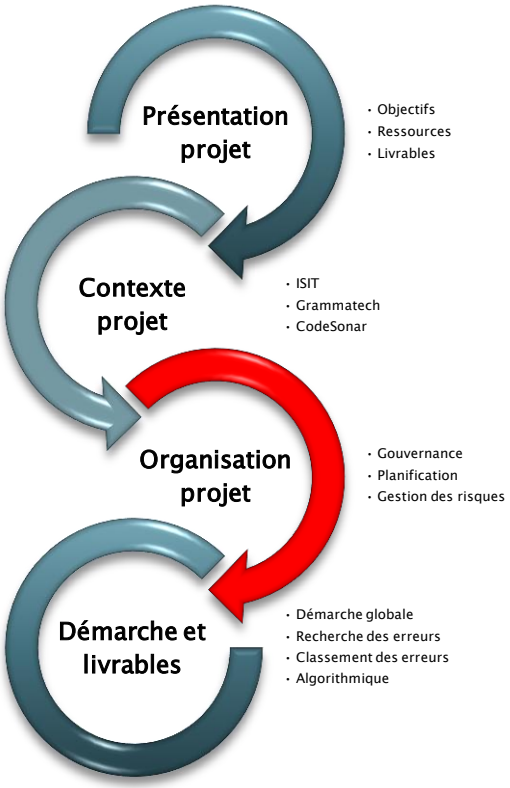
CodeSonar

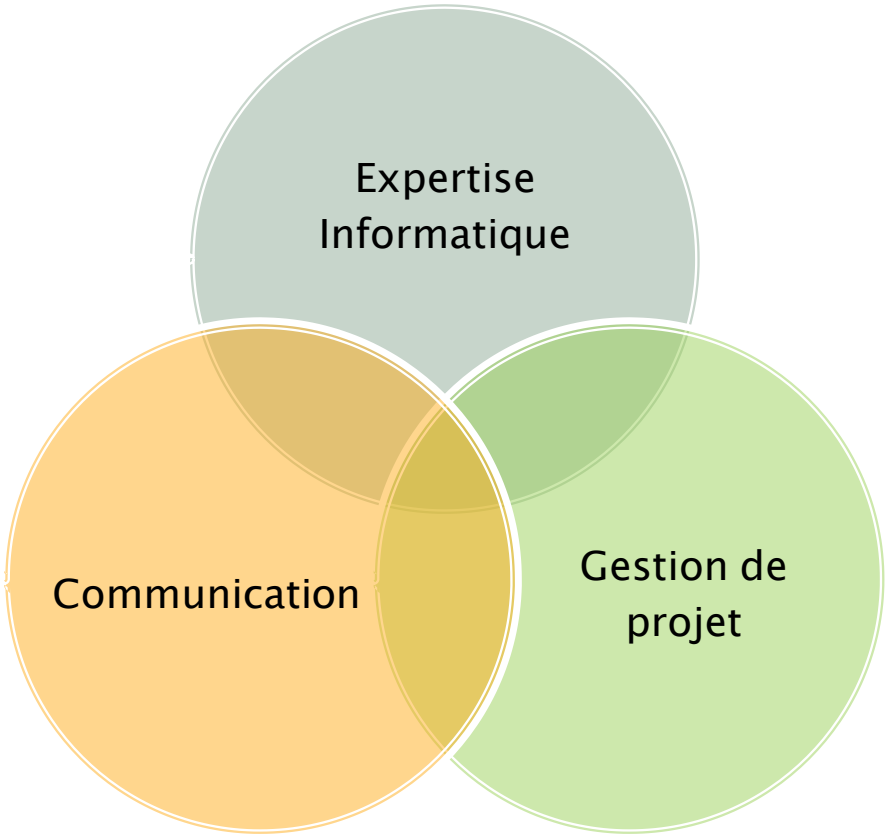

Présentation projet
Objectifs
Ressources
Livrables
Contexte projet
ISIT
Grammatech
CodeSonar
Organisation projet
Gouvernance
Planification
Gestion des risques
Démarche et livrables
Démarche globale
Recherche des erreurs
Classement des erreurs
Algorithmique


Defect found
PC-lint
C/C++
Syntax Checker
PolySpace
TECHNOLOGIES
klocwork
CODESONAR
coverity™
a higher code™
False positive

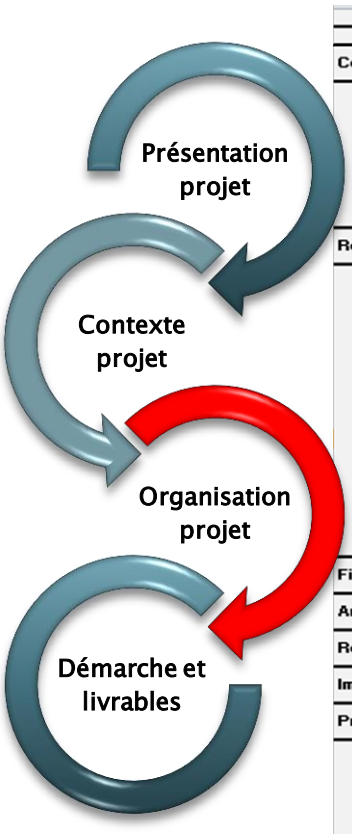

Présentation projet
Objectifs
Ressources
Livrables
Contexte projet
ISIT
Grammatech
CodeSonar
Organisation projet
Gouvernance
Planification
Gestion des risques
Démarche et livrables
Démarche globale
Recherche des erreurs
Classement des erreurs
Algorithmique

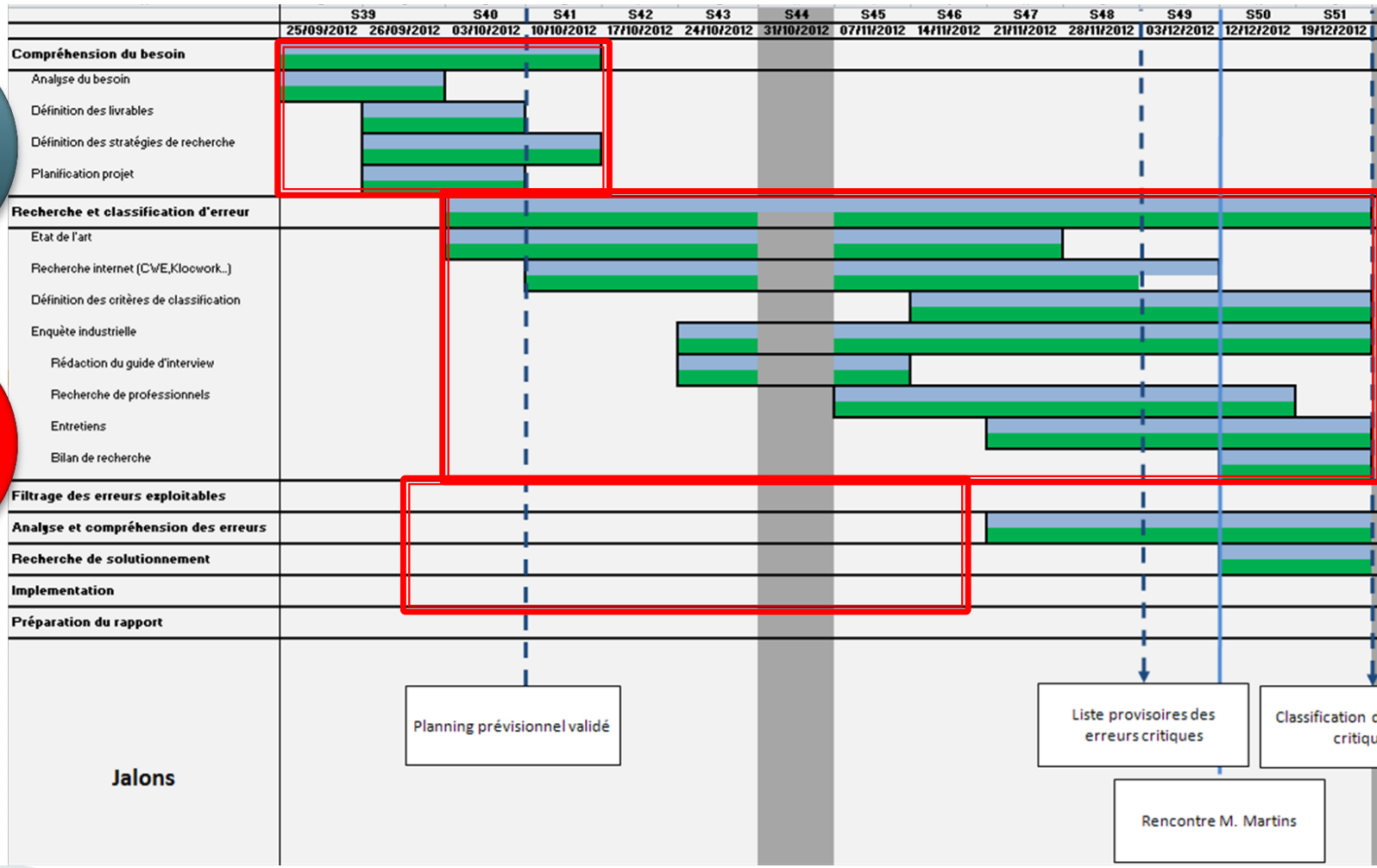
Gouvernance



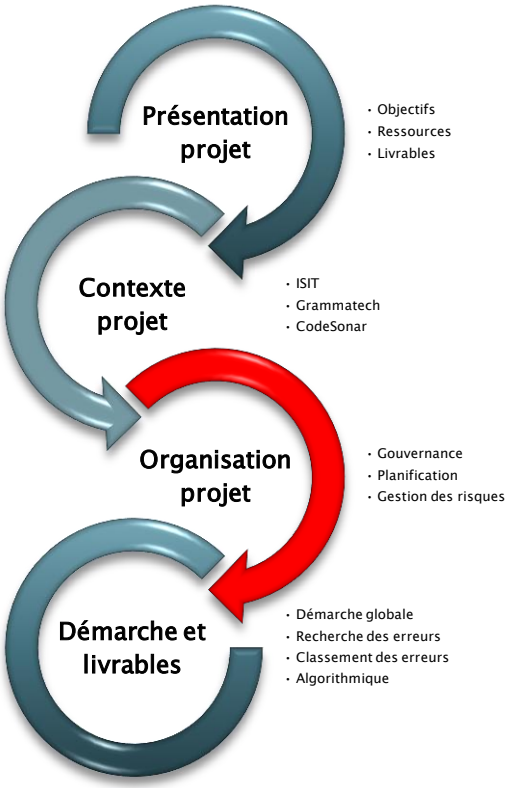


Planification



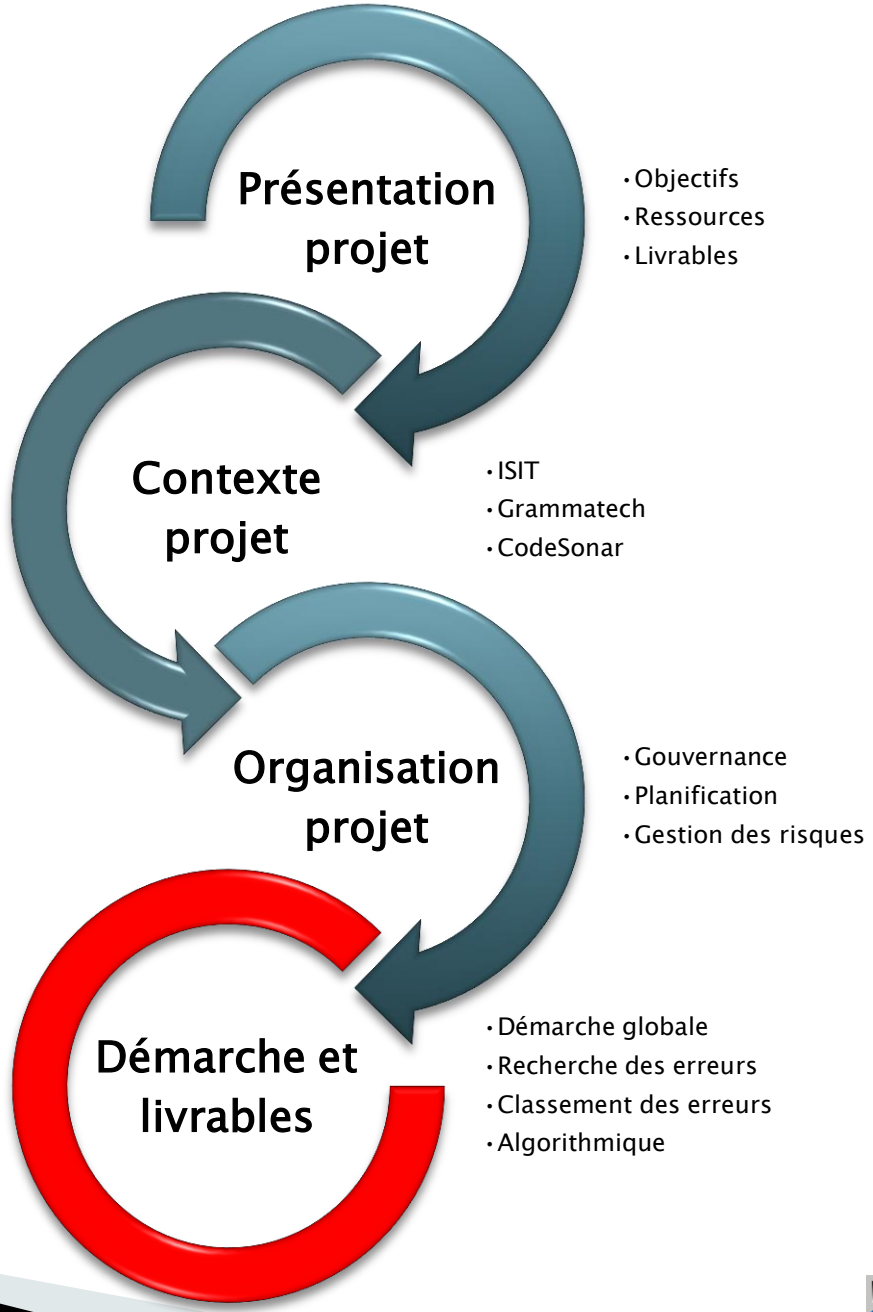


Gestion des risques

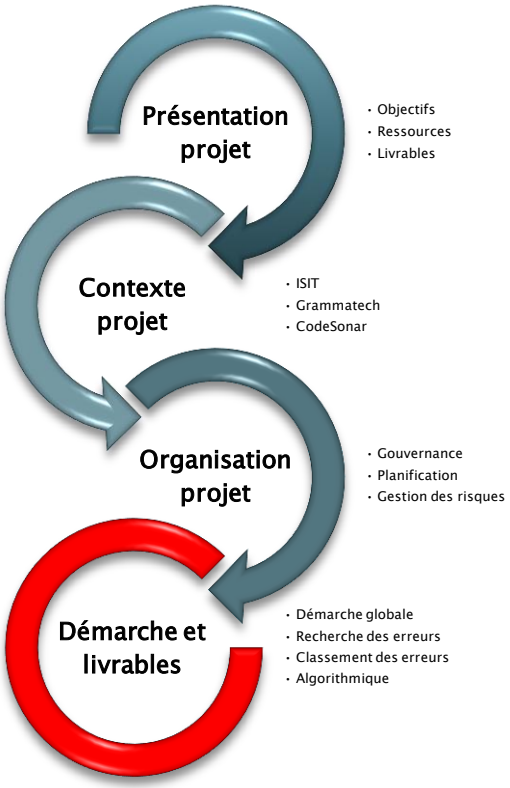


Risques	Gravité	Occurrence	Détection	Criticité
Non compréhension du sujet	3	2	1	6
Mauvaise définition des livrables	2	2	2	8
Retard sur la livraision des livrables	2	3	2	12
Non compréhension de la documentation technique	1	3	3	9
Abscence de pistes de recherche d'erreurs	3	3	1	9
Manque de communication avec les différentes parties prenantes	2	3	1	6
Non obtention du logiciel CodeSonar	2	3	1	6
Non fonctionnement du plugin	2	2	2	8
Aucune erreur implémentable trouvée	2	3	2	12
Mauvaise définition des pistes de recherche d'erreurs	2	3	2	12
Erreur etudiée déjà intégrée dans CodeSonar	2	3	3	18

	Niveau	Description
Gravité	1	Problème mineur
	2	Aboutissement partiel du projet
	3	Non aboutissement du projet
	4	Conséquence sur le logiciel
Occurrence	1	Tres peu probable
	2	Peu probable
	3	Probable
	4	Très probable
Détection	1	Facilement détectable
	2	Détectable
	3	Difficilement détectable


Présentation projet
Objectifs
Ressources
Livrables
Contexte projet
ISIT
Grammatech
CodeSonar
Organisation projet
Gouvernance
Planification
Gestion des risques
Démarche et livrables
Démarche globale
Recherche des erreurs
Classement des erreurs
Algorithmique

Démarche globale

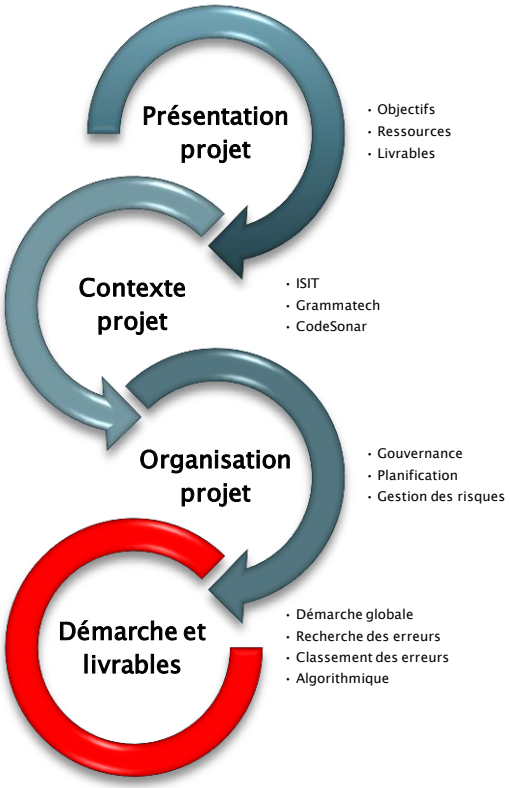


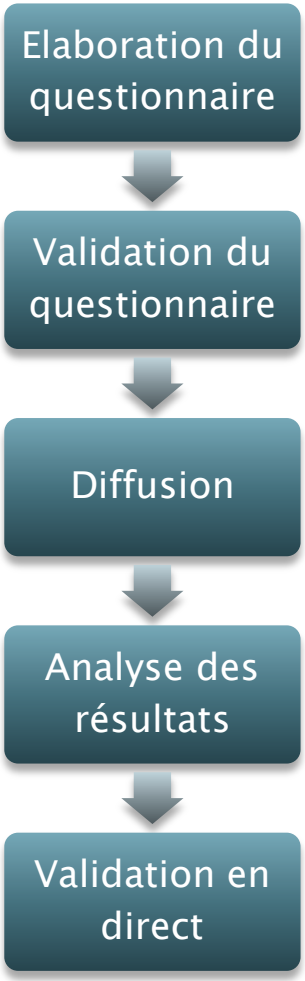
Recherche des erreurs
- Enquête industrielle
- Recherche internet
- Benchmark

Classement des erreurs
- Définition des critères
- Notation

Algorithmique
- Analyse de Code Sonar/API
- Réflexion d'implémentation

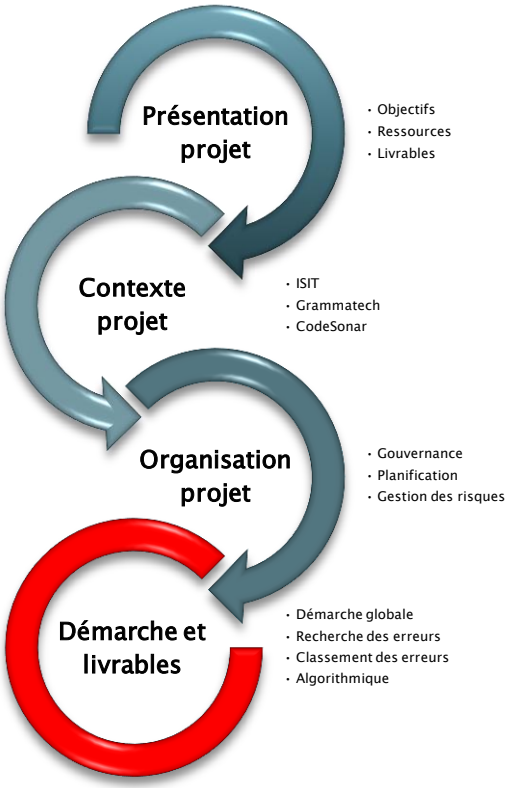
Enquête industrielle





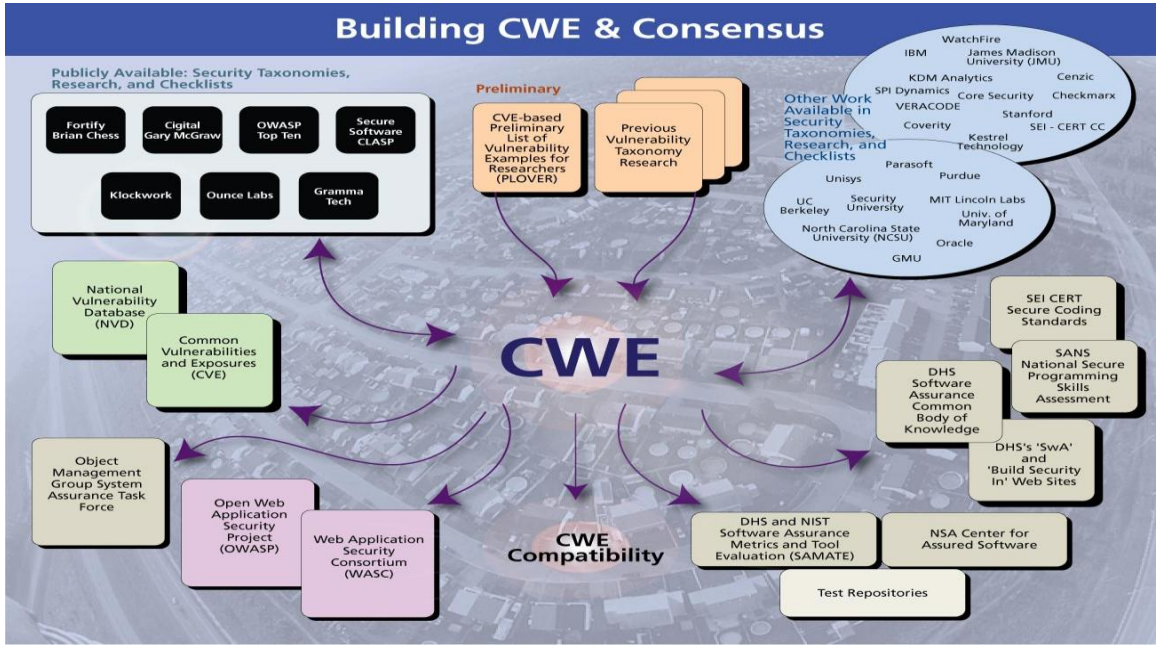
- Faible retour
  - Importance moindre chez les industriels
  - Problématique complexe

Recherche internet

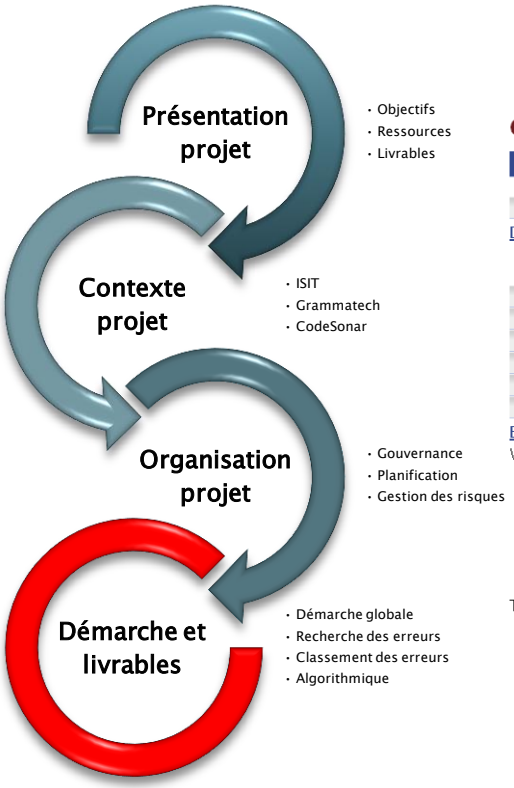


- Common Weakness Enumeration

MITRE



Recherche internet



CWE-121: Stack-based Buffer Overflow

Stack-based Buffer Overflow

Weakness ID: 121 *(Weakness Variant)* **Status:** Draft

Description

Description Summary

A stack-based buffer overflow condition is a condition where the buffer being overwritten is allocated on the stack (i.e., is a local variable or, rarely, a parameter to a function).

Alternate Terms
Time of Introduction
Applicable Platforms
Common Consequences
Likelihood of Exploit
Demonstrative Examples

Example 1

While buffer overflow examples can be rather complex, it is possible to have very simple, yet still exploitable, stack-based buffer overflows:

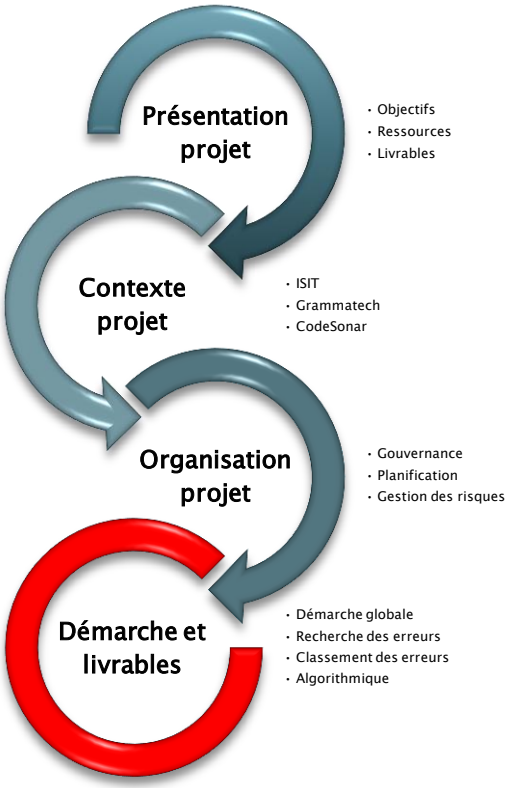
Example Language: **C** *(Bad Code)*

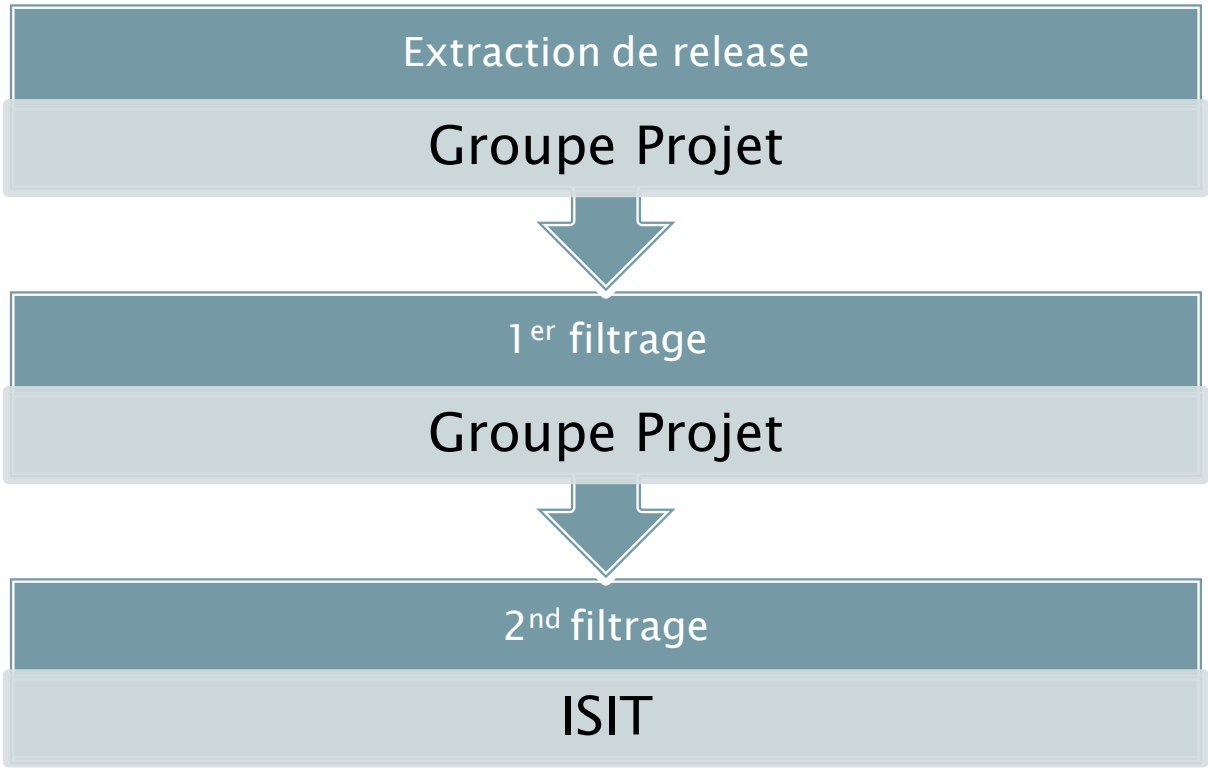
```
#define BUFSIZE 256
int main(int argc, char **argv) {
  char buf[BUFSIZE];
  strcpy(buf, argv[1]);
}
```

The buffer size is fixed, but there is no guarantee the string in argv[1] will not exceed this size and cause an overflow.

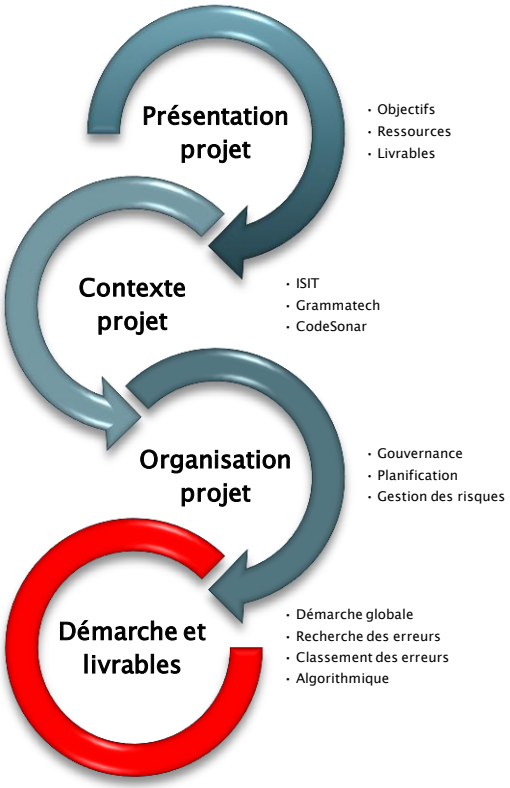
- Complexité d'utilisation
- Description des erreurs

Benchmark

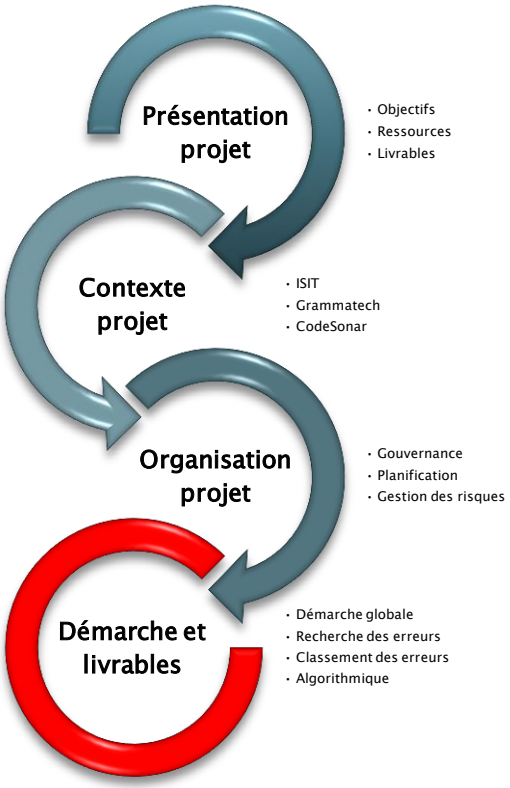

Présentation projet
Objectifs
Ressources
Livrables
Contexte projet
ISIT
Grammatech
CodeSonar
Organisation projet
Gouvernance
Planification
Gestion des risques
Démarche et livrables
Démarche globale
Recherche des erreurs
Classement des erreurs
Algorithmique


Extraction de release
Groupe Projet
1er filtrage
Groupe Projet
2nd filtrage
ISIT

Classement des erreurs



Classement des erreurs

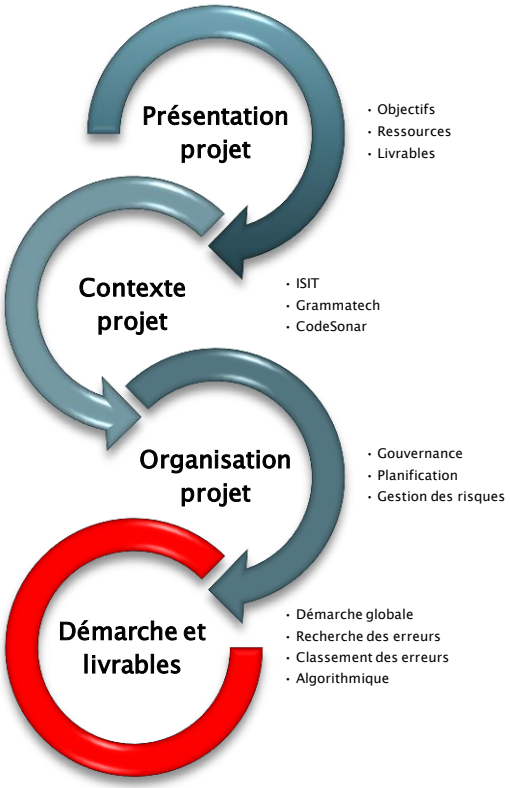


Faisabilite	
1	Manipulation de la mémoire
2	Utilisation des techniques de détection avancées
3	Utilisation de peu de technique de détection

Criticité	
1	Mineur
2	Moyen
3	Critique

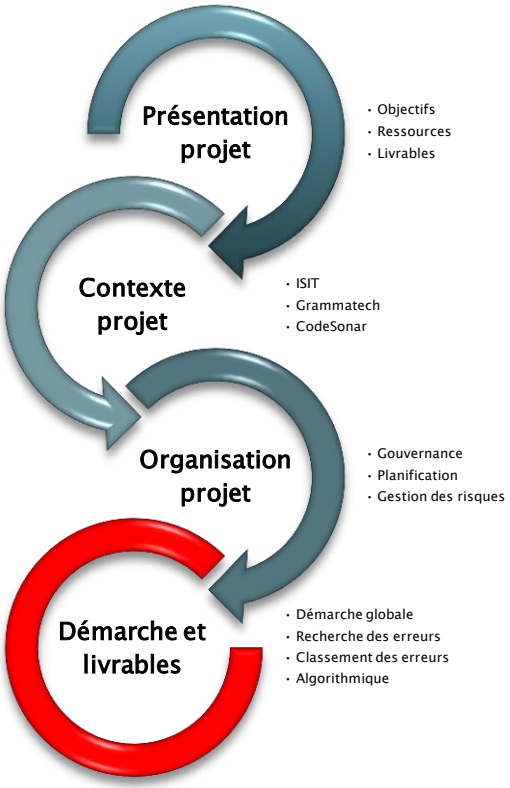
Facilité de détection	
1	Difficile
2	Moyen
3	Facile

Classement des erreurs



CWE ID	Dénomination	Criticité	Facilité de détection	Faisabilité	Classement
587	Affectation d'une adresse fixe à un pointeur	3	3	3	27
121	Stack-based Buffer Overflow	3	2	3	18
122	Heap-based Buffer Overflow	3	2	3	18
606	Donnée non vérifié pour une condition d'une boucle	2	3	3	18
467	L'utilisation de sizeof () sur un type pointeur	2	3	3	18
129	Improper Validation of Array Index	3	3	2	18
805	Buffer Access with Incorrect Length Value	2	3	2	12
362	ressources partagées avec synchronisation incorrecte («une	3	3	2	18
131	Incorrect Calculation of Buffer Size	3	3	2	18
682	Incorrect Calculation	3	2	2	12
390	la détection des erreurs sans action	3	1	2	6
197	Erreur troncature numérique	2	1	3	6
135	Calcul incorrect de la longueur d'une chaine de caractères multi-octet	2	1	3	6
20	Improper input validation	3	1	2	6

Algorithmique



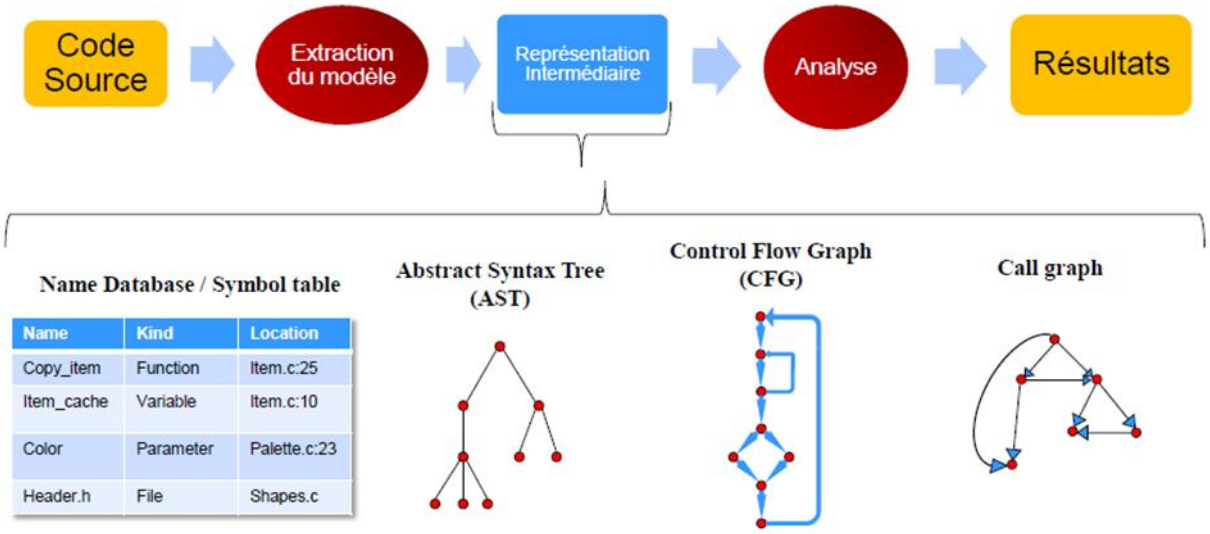
- Objectifs
- Ressources
- Livrables

- ISIT
- Grammatech
- CodeSonar

- Gouvernance
- Planification
- Gestion des risques

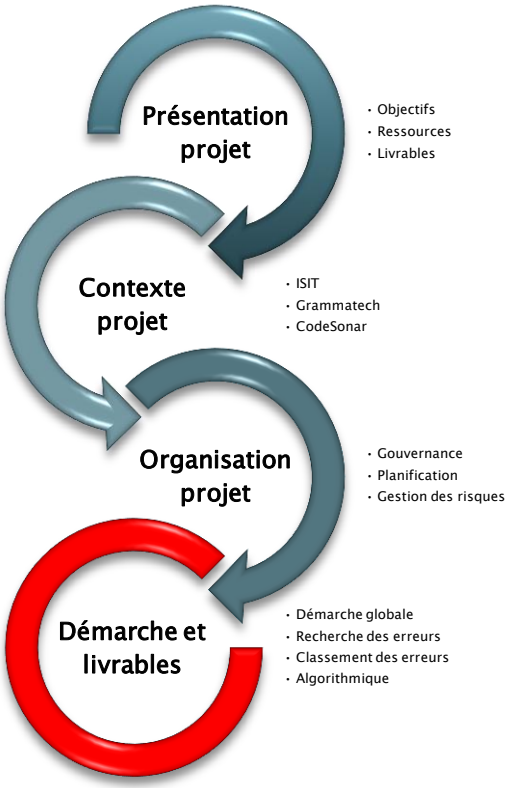
- Démarche globale
- Recherche des erreurs
- Classement des erreurs
- Algorithmique

Fonctionnement de CodeSonar



Name	Kind	Location
Copy_item	Function	Item.c:25
Item_cache	Variable	Item.c:10
Color	Parameter	Palette.c:23
Header.h	File	Shapes.c

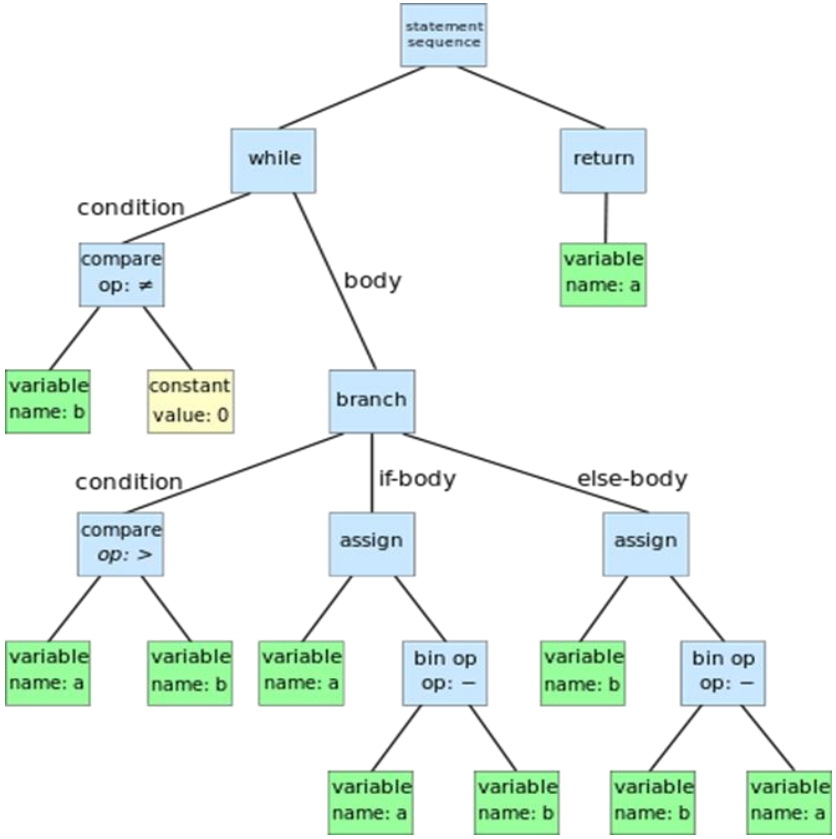
Algorithmique



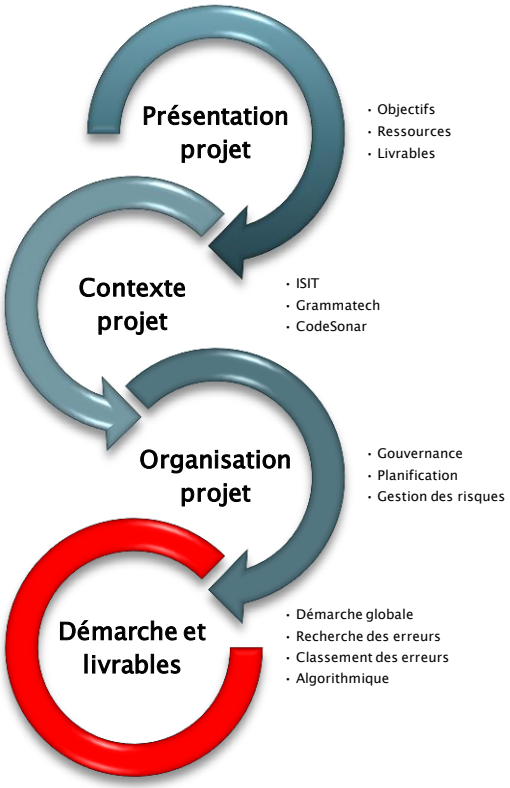
Abstract Syntax Tree

```
while b ≠ 0
  if a > b
    a := a − b
  else
    b := b − a
return a
```

Algorithme d'Euclide



Algorithmique



Nom d'erreur : Stack-based Buffer Overflow

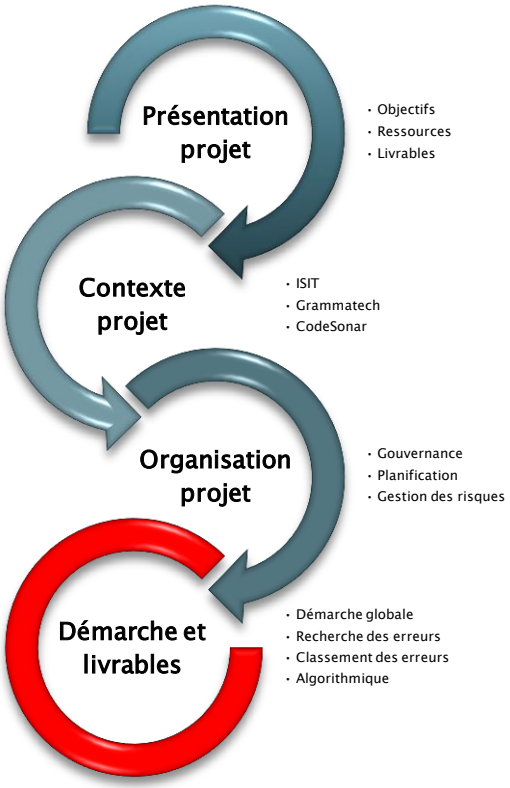
Numéro d'erreur : CWE - 121

Description :

A stack-based buffer overflow condition is a condition where the buffer being overwritten is allocated on the stack.

```
1   #include <stdio.h>
2   #include <string.h>
3   int main()
4   {
5   char chaine_1[10];
6   char chaine_2[15];
7   int i=1;
8   while (i<=2)
9   {
10  printf("Entrer un mot: ");
11  gets(chaine_2);
12  if (sizeof(chaine_1)< strlen(chaine_2))
13  strcpy(chaine_1, chaine_2);
14  i++;
15  }
16  return 0;
17  }
18
```

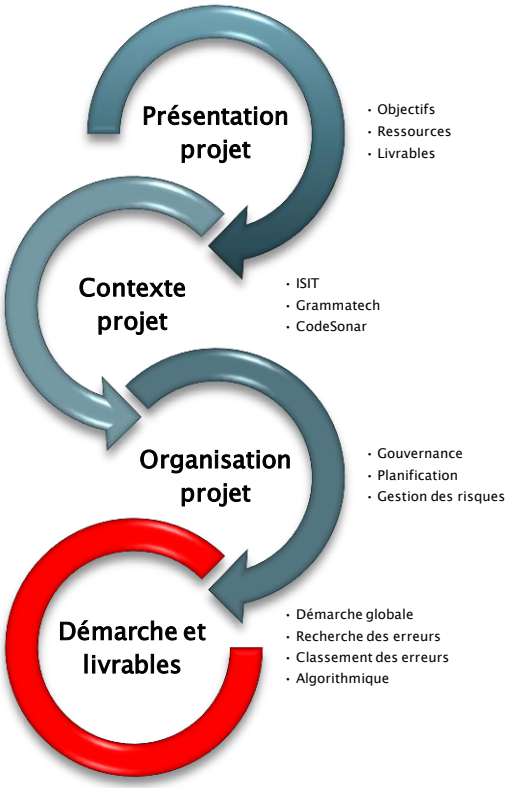
Algorithmique



Algorithme de détection de l'erreur 121 et 122 :

Pseudo Code	Solution d'implémentation (API CS)
1. Début	
2. Créer Abstract Syntax Tree du code	cs_abs_loc ast(a,from,outast)
3. Chercher les utilisations de strcpy	(abs-loc-lookup-by-name strcpy) : ABS_LOC | #f
Lieux ← Endroits d'utilisation de strcpy	create_structure()
4. Tant que Lieux ≠ Vide	
Faire	
4.1.Lieu ← Un élément de Lieux (une utilisation de strcpy)	
4.2.Nœud ← Nœud parent de Lieu	
4.3.Si Nœud ≠ Branch (n'est pas une branche conditionnelle)	cs_ast_iter_first
4.3.1. Afficher "Absence de comparaison" devant strcpy.	cs_report_warning
4.3.2. Aller à 4.7.	
4.4.Fin Si	
4.5.Si Mauvaise_comparaison(Branch, Lieu)	
4.5.1. Afficher "Mauvaise comparaison" devant strcpy.	cs_report_warning
4.6.Fin Si	
4.7.Retirer Lieu de Lieux	
5. Fin Tant que	
6. Quitter	
Bool Fonction Mauvaise_comparaison(Branch, Lieu)	
1. Début	
2. Si (Branch.compare_op == '>' OR '>=' OR '==')	cs_ast_iter_next
2.1. Si (variable_name_1 ≠ sizeof(Lieu.variable_name_1) OR variable_name_2 ≠ strlen(Lieu.variable_name_2))	
2.1.1 Return True.	
2.2. Fin Si	
3. Fin Si	
4. Si (Branch.compare_op == '<' OR '<=')	cs_ast_iter_next
4.1. Si (variable_name_1 ≠ strlen(Lieu.variable_name_2) OR variable_name_2 ≠ sizeof(Lieu.variable_name_1))	
4.1.1 Return True.	
4.2. Fin Si	
5. Fin Si	
6. Return False	

Algorithmique



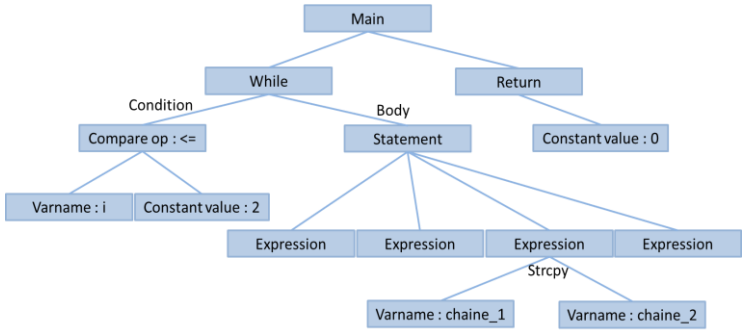
- Objectifs
- Ressources
- Livrables

- ISIT
- Grammatech
- CodeSonar

- Gouvernance
- Planification
- Gestion des risques

- Démarche globale
- Recherche des erreurs
- Classement des erreurs
- Algorithmique

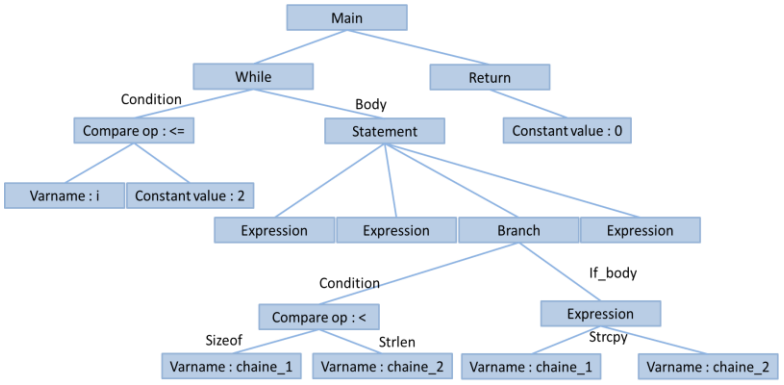
Test 1:




```c
#include <stdio.h>
#include <string.h>
int main()
{
char chaine_1[10];
char chaine_2[15];
int i=1;
while (i<=2)
{
printf("Entrer un mot: ");
gets(chaine_2);
if (sizeof(chaine_1)< strlen(chaine_2))
strcpy(chaine_1, chaine_2);
i++;
}
return 0;
}
```

Sortie : warning

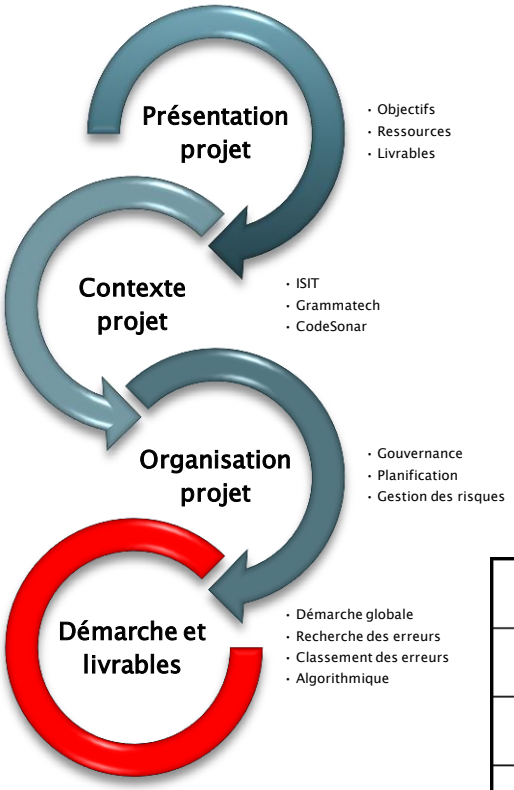
Test 2:




```c
#include <stdio.h>
#include <string.h>
int main()
{
char chaine_1[10];
char chaine_2[15];
int a=1;
while (a<=5)
{
printf("Entrer un mot: ");
gets(chaine_2);
strcpy(chaine_1, chaine_2);
a++;
}
return 0;
}
```

Sortie : warning

Algorithmique



Avancement	
0	Non commencée
1	Analysée
2	En cours
3	Solutionnée
4	Développée

CWE ID	Dénomination	Criticité	Facilité de détection	Faisabilité	Classement	Avancement
587	Affectation d'une adresse fixe à un pointeur	3	3	3	27	3
121	Stack-based Buffer Overflow	3	2	3	18	3
122	Heap-based Buffer Overflow	3	2	3	18	3
606	Donnée non vérifié pour une condition d'une boucle	2	3	3	18	1
467	L'utilisation de sizeof () sur un type pointeur	2	3	3	18	3

Conclusion

- Contraintes:
  - L'éloignement géographique avec la société ISIT
  - La complexité du sujet par rapport à notre niveau de connaissance en informatique
  - L'abondance de la documentation liée à l'utilisation et à la programmation de plugin pour CodeSonar

- Actions:
  - Formation faite pour faciliter l'utilisation du logiciel Code Sonar
  - Encadrement par M MARTINS et M TODOSKOFF

Merci de votre attention

Bibliographie

Webographie

- *http://www.isit.fr/*
- *http://cwe.mitre.org/*
- *http://grammatech.com*

Bibliographie

- *CodeSonar Plugin Development Guide_V0.1, Haoxian Howard Wang, 25/09/2012*
- *Formation Codesonar – version 5.0, ISIT, 2010*
- *Documentation technique CodeSonar ISIT/Grammatech*
- *Audit de code & comparaison d'outils d'analyse statique, HUIBAN Benoit MAHJOUBI Siham MESSAOUDI Youness, 2011-2012*
- *A Comparative Study of Industrial Static Analysis Tools, Emanuelsson and Ulf Nilsson, January 7, 2008*