SOFTWARE DEFECT PREDECTION

In [85]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
import itertools
```

In [86]:

```
data=pd.read_csv("Software Defect Dataset.csv")
```

In [87]:

```
data.head()
```

Out[87]:

	loc	v(g)	ev(g)	iv(g)	n	v	l	d	i	e	...	lOCode	lOCom
0	1.1	1.4	1.4	1.4	1.3	1.30	1.30	1.30	1.30	1.30	...	2	
1	1.0	1.0	1.0	1.0	1.0	1.00	1.00	1.00	1.00	1.00	...	1	
2	72.0	7.0	1.0	6.0	198.0	1134.13	0.05	20.31	55.85	23029.10	...	51	
3	190.0	3.0	1.0	3.0	600.0	4348.76	0.06	17.06	254.87	74202.67	...	129	
4	37.0	4.0	1.0	4.0	126.0	599.12	0.06	17.19	34.86	10297.30	...	28	

5 rows × 22 columns

In [88]:

```
data.describe()
```

Out[88]:

	loc	v(g)	ev(g)	iv(g)	n	v
count	10885.000000	10885.000000	10885.000000	10885.000000	10885.000000	10885.000000
mean	42.016178	6.348590	3.401047	4.001599	114.389738	673.758017
std	76.593332	13.019695	6.771869	9.116889	249.502091	1938.856196
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	11.000000	2.000000	1.000000	1.000000	14.000000	48.430000
50%	23.000000	3.000000	1.000000	2.000000	49.000000	217.130000
75%	46.000000	7.000000	3.000000	4.000000	119.000000	621.480000
max	3442.000000	470.000000	165.000000	402.000000	8441.000000	80843.080000

In [89]:

```
data.shape
```

Out[89]:

```
(10885, 22)
```

In [90]:

```
data.isnull().sum()
```

Out[90]:

```
loc                 0
v(g)                0
ev(g)               0
iv(g)               0
n                   0
v                   0
l                   0
d                   0
i                   0
e                   0
b                   0
t                   0
lOCode              0
lOComment           0
lOBlank             0
locCodeAndComment   0
uniq_Op             0
uniq_Opnd           0
total_Op            0
total_Opnd          0
branchCount         0
defects             0
dtype: int64
```

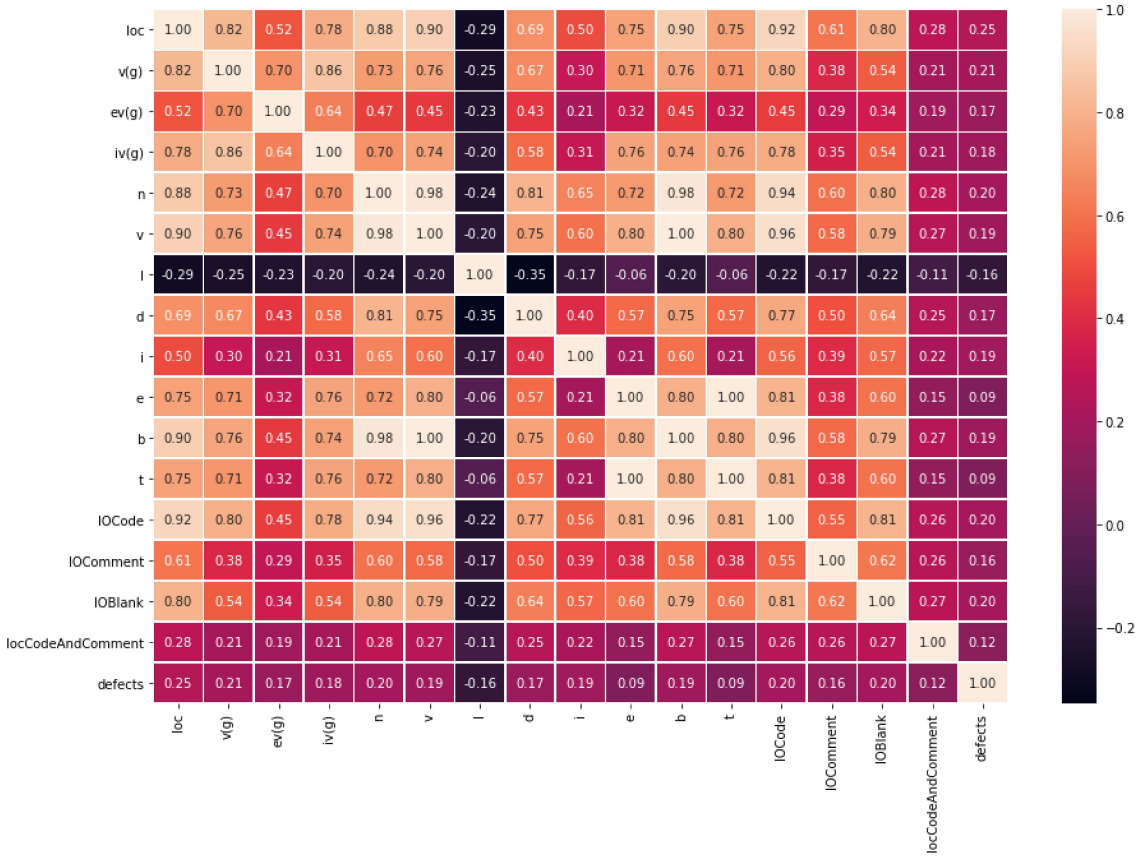
In [91]:

```
defects_true_false = data.groupby('defects')['b'].apply(lambda x: x.count()) #defect rat
print('False : ' , defects_true_false[0])
print('True : ' , defects_true_false[1])
```

```
False :  8779
True :  2106
```

In [92]:

```
f,ax = plt.subplots(figsize = (15, 10))
sns.heatmap(data.corr(), annot = True, linewidths = .5, fmt = '.2f')
plt.show()
```



In [93]:

```
def plot_confusion_matrix(cm, classes,
    normalize=False,
    title='Confusion matrix',
    cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
        thresh = cm.max() / 2.
        for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")
        plt.tight_layout()
        plt.ylabel('True label')
        plt.xlabel('Predicted label')
```

In [94]:

```
x = data.iloc[:, :-10].values  #Select related attribute values for selection
y = data.defects   #Select classification attribute values
x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size = 0.
classifier = GaussianNB()
classifier.fit(x_train, y_train)
pred_train=classifier.predict(x_train)
pred_test=classifier.predict(x_test)
```

In [95]:

```
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(classifier, x, y, cv=10, scoring ='
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.19832338079926504
Variance is:  0.1809830041341295
Accuracy is:  0.8190169958658705
Cross Validation result is:  0.8044157315129908
Confusion matrix, without normalization
              precision    recall  f1-score   support

       False       0.83      0.98      0.90      1774
        True       0.55      0.13      0.21       403

    accuracy                           0.82      2177
   macro avg       0.69      0.55      0.55      2177
weighted avg       0.78      0.82      0.77      2177
```