Verifysoft TECHNOLOGY

Integrated Systems Development Limited

# Testwell CTC++ User Testimonial

## ISD: Testwell CTC++ gives us what we need while keeping the process simple

Integrated Systems Development (Pvt) Ltd (ISD) is a medical device development company that design, patent and market laboratory equipment for Assisted Reproductive Technologies (ART).

ART is used to help infertile couples to have a baby of their own by fertilization of oocytes with sperm from the partner outside the body, a procedure within ART known as In Vitro Fertilization (IVF).

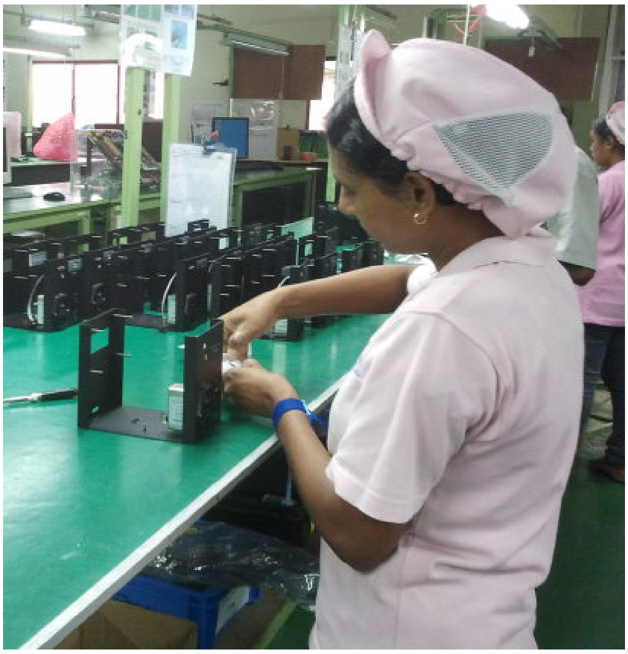
Production site at ISD, Sri Lanka

The sperm and recovered oocytes are mixed in a dish to create embryos that are transferred back to the uterus. The demand for more advanced treatments use more sophisticated equipment, consumables and culture media to further optimize the outcome.

Even though IVF has been around for a while, the safety regulatory bodies have been relatively relaxed regarding equipment on the market. It is only in recent times that the regulatory authorities have tightened the requirements. Most of the equipment in the market are certified under IEC61010 as laboratory equipment. However, equipment manufacturers have started to certify their equipment under IEC60601 as "Medical Devices". According to the European Union, medical devices are classified into 3 classes according to the risk levels with class 3 bearing the highest risk.

ISD's equipment will fall under class 2 as medium risk devices. When it comes to software/firmware there are no clear guidelines. Although IEC62304 which is the standard for medical device software exists there are no specific requirements which can be derived directly.

According to section 5 of the standard unit implementation and verification, integration testing and system testing are important components of the software development process.



Our standards are very much higher than the current industry norm. The firmware in our product controls the temperature which is critical to IVF. Numerous safety checks on the integrity of the entire system are dependent on the correct firmware. Black box tests verify all the product requirements. But this does not guarantee that there are no defects. White box testing methods need to be employed for verification.

Since the modern day products have sophisticated firmware in them, it is nearly impossible to cover the entire code without knowing the conditions. We need tools that would help us to identify and execute these conditions.

**We chose Testwell CTC++** in order to aid us in the unit tests and verification. Being a typical bare metal embedded system without a file system, counters are needed to store information and later downloaded from the target to source. **One of the biggest challenges that we had was the limited memory in our systems. Testwell CTC++ required a very small footprint for its instrumentation and the Host target add-on makes integrations to any system easy.** The important point here is that **we don't have any restrictions with compilers, IDEs or debug tools.** We are able to use Testwell CTC++ without changing our development process drastically. The reports are presented in HTML. They are simple and intuitive to read and understand. It makes it easier for the management to visualize the results and to understand what has been done. It is easier to browse through each module separately and browse annotated code which makes the whole process simpler. These reports will be submitted to regulatory authorities as a part of testing and verification data.

| TER % - multicondition | | TER % - statement | | File |
|---|---|---|---|---|
| **Directory: ../.** | | | | |
| 100 % | (0/0) | 100 % | (0/0) | CTCArray.c |
| 76 % - | (259/341) | 86 % - | (497/578) | main.cpp |
| **76 % -** | (259/341) | **86 % -** | (497/578) | **DIRECTORY OVERALL** |
| **Directory: ../../../../03 - Modules/alphanumeric_display/alphanumeric_display/alphanumeric_display** | | | | |
| 95 % - | (115/121) | 99 % - | (244/246) | alphanumeric_display.cpp |
| **95 % -** | (115/121) | **99 % -** | (244/246) | **DIRECTORY OVERALL** |
| **Directory: ../../../../03 - Modules/debug_print/debug_print/debug_global** | | | | |
| 30 % - | (14/46) | 28 % - | (15/53) | debug_global.c |
| **30 % -** | (14/46) | **28 % -** | (15/53) | **DIRECTORY OVERALL** |
| **Directory: ../../../../03 - Modules/key_pad/key_pad/key_pad** | | | | |
| 41 % - | (31/76) | 46 % - | (56/121) | key_pad.cpp |
| **41 % -** | (31/76) | **46 % -** | (56/121) | **DIRECTORY OVERALL** |
| **Directory: ../../../../03 - Modules/timer/timer/timer** | | | | |
| 92 % - | (11/12) | 100 % | (17/17) | timer.cpp |
| **92 % -** | (11/12) | **100 %** | (17/17) | **DIRECTORY OVERALL** |

Testwell CTC++ gives statement coverage, branch coverage and most crucially MCDC (Modified condition/ Decision coverage). While the first two are relatively straightforward important, it is MCDC which picks out the evilest bugs in the code. Compound IF statements with multiple conditions are never intuitive and its one of the biggest source of bugs. **Testwell CTC++ makes it ever easier to weed out these bugs by pointing to the right test condition.**

```
  1       188 int8_t Tsic506F::readSens(uint16_t &temp_value){
          189
          190     /* returnValue will be modified to TSIC506_TIMEOUT if an error occurs*/
          191     int8_t returnValue=TSIC506_SUCCESSFUL;
          192
          193     /* Max value for timeout is defines as TIMEOUT */
          194     uint16_t timeout = 0;
          195
          196     /* wait until start bit starts*/
  1     1 197     while (TSIC HIGH && (timeout < TIMEOUTVALUE))
  1       197      1: T && (T)
        0 197      2: T && (F)
        1 197      3: F && (_)
          198     {
          199         timeout++;
          200         _delay_us(10);
          201     }
          202     /*No time out */
  1     0 203     if(timeout<TIMEOUTVALUE)
          204     {
          205         uint16 t strobelength = 0;
          206         timeout = 9900;
          207         /* Measure Strobe time */
          208         /* wait for rising edge */
  1     2 209         while (TSIC LOW && (timeout < TIMEOUTVALUE))
  1       209         1: !(F) && (T)
        1 209         2: !(T) && ( )
        1 209         3: !(F) && (F)
          210         {
          211             strobelength++;
          212             timeout++;
          213             _delay_us(10);
          214         }
```

**Testwell CTC++**

```
            206   if((setPoint >= ALLOWABLE_SP_MIN) && (setPoint <= ALLOWABLE_SP_MAX) && (setPointMax > setPoint) && \
  1     6   207     (setPoint > setPointMin) && (calibFactor >= ALLOWABLE_CF_MIN) && (calibFactor <= ALLOWABLE_CF_MAX))
  1         207    1: (T) && (T) && (T) && (T) && (T) && (T)
        1   207    2: (T) && (T) && (T) && (T) && (T) && (F)
        1   207    3: (T) && (T) && (T) && (T) && (F) && (_)
        1   207    4: (T) && (T) && (T) && (F) && (_) && (_)
        1   207    5: (T) && (T) && (F) && (_) && (_) && (_)
        1   207    6: (T) && (F) && (_) && (_) && (_) && (_)
        1   207    7: (F) && (_) && (_) && (_) && (_) && (_)
            208   {
            209     /* set glass state */
            210     resetGlass();
            211     return_value = HEATER_CONTROLLER_SUCCESSFUL;
```

The above example is a critical part of our code where key decisions are taken. A black box test may not have tested all the above scenarios, let alone identify them.

Testwell CTC++ was easy to setup. The compilers we use were not even listed on the site. But when we requested for a free evaluation, the support they offered was refreshingly positive and we **were able to set up everything and obtain the coverage report within a day.** Being a small company, we run on a small budget. Verifysoft's licensing method was flexible enough to convince the management that the product was worth investing in. Before obtaining Testwell CTC++ we looked at other tools which were harder to integrate and had limitations when it came to instrumentation for small targets.

In conclusion, Testwell CTC++ gives us what we need while keeping the process simple.

Integrated Systems Development Limited

Integrated Systems Development (Pvt) Ltd (ISD)
Mr. Harin de Silva
56/9 Kesbewa Road
Boralesgamuwa, Colombo 10290
Sri Lanka
www.isd-limited.com


For further questions please visit www.verifysoft.com and contact us at +49 781 127 8118-0